Minimal Tail-Biting Trellises: The Golay Code and More

A. R. Calderbank, *Fellow, IEEE*, G. David Forney, Jr., *Fellow, IEEE*, and Alexander Vardy, *Fellow, IEEE*

dedicated to Gus Solomon, who pioneered this field

***Abstract*—Tail-biting trellis representations of block codes are investigated. We develop some elementary theory, and present several intriguing examples, which we hope will stimulate further developments in this field. In particular, we construct a 16-state 12-section structurally invariant tail-biting trellis for the $(24, 12, 8)$ binary Golay code. This tail-biting trellis representation is minimal: it simultaneously minimizes all conceivable measures of state complexity. Moreover, it compares favorably with the minimal conventional 12-section trellis for the Golay code, which has 256 states at its midpoint, or with the best quasi-cyclic representation of this code, which leads to a 64-state tail-biting trellis. Unwrapping this tail-biting trellis produces a periodically time-varying 16-state rate-$1/2$ "convolutional Golay code" with $d = 8$, which has attractive performance/complexity properties. We furthermore show that the $(6, 3, 4)$ quaternary hexacode has a minimal 8-state group tail-biting trellis, even though it has no such linear trellis over $\mathbb{F}_4$. Minimal tail-biting trellises are also constructed for the $(8, 4, 4)$ binary Hamming code, the $(4, 2, 3)$ ternary tetracode, the $(4, 2, 3)$ code over $\mathbb{F}_4$, and the $\mathbb{Z}_4$-linear $(8, 4, 4)$ octacode.**

***Index Terms*—Convolutional codes, Golay code, hexacode, iterative decoding, tail-biting, trellises.**

I. Introduction

TRELLIS representations of linear block codes have received much attention in recent years (cf. special issue [14] on "Codes and Complexity" of the IEEE TRANSACTIONS ON INFORMATION THEORY). Trellis representations not only illuminate code structure, but also often lead to efficient trellis-based decoding algorithms.

It is now well known [17], [50], [64] that, given a coordinate ordering, there exists a unique (up to isomorphism) minimal conventional trellis for any linear block code, or indeed for any group code [22]. The minimal trellis simultaneously minimizes all of the usual measures of trellis complexity, such as state complexity, branch complexity, or Viterbi decoding complexity [46], [64].

Manuscript received August 8, 1997; revised November 20, 1998. This work was supported by the Packard Foundation and the National Science Foundation. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, MIT, Cambridge, MA, August 16–21, 1998.

A. R. Calderbank is with the AT&T Shannon Laboratories, Florham Park, NJ 07932 USA.

G. D. Forney, Jr. is with Motorola, Inc., Mansfield, MA 02048 USA.

A. Vardy is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA.

Communicated by F. R. Kschischang, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(99)04735-5.

This paper considers unconventional "tail-biting" trellises, in which the "time axis" has the topology of a circle rather than that of an interval [57]. We will develop some basic theory of tail-biting trellis representations, construct specific tail-biting trellises for several remarkable block codes, and show that these trellises are minimal. Along the way, we will encounter a few surprises. In particular, we will observe that the minimal possible tail-biting trellis for a linear code $\mathbb{C}$ can have fewer states if $\mathbb{C}$ is regarded as a *group* code rather than as a *linear* code (namely, a group trellis for $\mathbb{C}$ can have fewer states than any linear trellis for $\mathbb{C}$). We will also find that "unwrapping" minimal tail-biting trellises for good block codes may often lead to time-varying convolutional codes with excellent performance/complexity tradeoff.

The $(24, 12, 8)$ binary Golay code $\mathbb{C}_{24}$ is arguably the most remarkable binary block code. It has often been used as a benchmark in studies of code structure and decoding algorithms. A simple argument of Muder [50] shows that a conventional trellis for $\mathbb{C}_{24}$ must have at least 256 states at its midpoint. This bound is met by the minimal trellis for $\mathbb{C}_{24}$ in the coordinate ordering of [17].

It has long been known that certain block codes have quasi-cyclic, or double-circulant [45, p. 505], representations which lead to tail-biting trellises with fewer states than any conventional trellis. For example, Solomon and van Tilborg [57] found a 64-state quasi-cyclic representation for $\mathbb{C}_{24}$.

A result of Wiberg, Loeliger, and Kötter [68], rederived in the next section, implies that the number of states 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. For example, a tail-biting trellis for $\mathbb{C}_{24}$ could have as few as $\sqrt{256} = 16$ states. One of the main results of this paper is that, in fact, there exists such a minimal tail-biting trellis for $\mathbb{C}_{24}$. This trellis will be constructed from a generator matrix for $\mathbb{C}_{24}$ which has the structure shown in (1) at the top of the following page, where $\star$ stands for either 1 or 0. The existence of such a generator matrix for the Golay code was conjectured by the second author [20] during the Allerton Conference in 1996. We will exhibit a specific generator matrix of this type in Section III. We will furthermore show that this matrix yields a "rate-$1/2$" 12-section tail-biting trellis for $\mathbb{C}_{24}$ with only 16 states.

In a companion paper [9], we classify all such generator matrices, using the Miracle Octad Generator representation of the Golay code. We show that a generator matrix of the

$$\begin{bmatrix}
\star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star \\
\star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star & \star & \star \\
\star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star & \star & \star \\
\star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star & \star & \star \\
\star & \star & \star & \star & \star & \star & \star & \star & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \star & \star
\end{bmatrix} \tag{1}$$

form (1) is unique up to symmetries of the Mathieu group M_{24}, which is the automorphism group of $\mathbb{C}_{24}$.

Similarly, a minimal conventional trellis for the $(6,3,4)$ hexacode H_6 over the field $\mathbb{F}_4$ has state-complexity profile $\{1,4,16,64,16,4,1\}$. We show that the best possible linear (over $\mathbb{F}_4$) tail-biting trellis for the hexacode necessarily contains state spaces of size 16. However, in Section IV, we exhibit a group (over the additive group of $\mathbb{F}_4$) tail-biting trellis for the hexacode with the minimal state-complexity profile $\{8,8,8,8,8,8\}$. This result is interesting and rather surprising, since the minimal conventional trellis for a linear code is always linear. Evidently, minimizing tail-biting trellises involves even more "art" than minimizing conventional trellises.

We also consider tail-biting trellis representations for several other short rate-$1/2$ self-dual codes. In particular, in Section V, we find:

- a $2/4$-state tail-biting trellis for the $(8, 4, 4)$ binary extended Hamming code;
- a 3-state tail-biting trellis for the $(4, 2, 3)$ tetracode over $\mathbb{F}_3$;
- a 2-state group tail-biting trellis for the $(2, 1, 2)$ code over $\mathbb{F}_4$;
- a 4-state group tail-biting trellis for the $(4, 2, 3)$ quadracode over $\mathbb{F}_4$;
- an $8/16$-state tail-biting trellis for the $(8, 4, 4)$ octacode over $\mathbb{Z}_4$.

These trellises are all shown to be minimal: their minimality follows from the square-root lower bound of [68] as well as another bound based on total generator span.

We further consider "unwrapping" the tail-biting generators to obtain convolutional or trellis codes. For example, by unwrapping the $\mathbb{C}_{24}$ generators in (4), we obtain a 16-state periodically time-varying self-dual rate-$1/2$ binary linear "Golay convolutional code," with period 4 and minimum distance $d = 8$. This improves on the value $d = 7$ for the best possible 16-state time-invariant rate-$1/2$ binary linear convolutional code. We conjecture that many other good convolutional codes can be found by a similar unwrapping of tail-biting generators for good block codes.

Finally, we briefly discuss decoding algorithms in Section VI. Tail-biting trellises are a nice testbed for the study of decoding algorithms for codes defined on graphs with cycles (cf. [68], [67], [20]), since their cycle structure is the simplest possible. There is also a considerable amount of literature on the performance and complexity of various decoding algorithms for $\mathbb{C}_{24}$ and H_6, which might be useful for comparisons.

II. Tail-Biting Trellises

In this section, we first define both conventional and tail-biting trellises. We then show how a set of generators for a linear or group block code may be used to construct a tail-biting trellis. We use the $(2, 1, 2)$ repetition code over $\mathbb{F}_4$ as an example, which illustrates several subtle points in this construction. Finally, we discuss the square-root bound of [67], [68], which is a special case of a general lower bound on state-space size for codes defined on factor graphs.

A. Definition of Tail-Biting Trellises

We begin by defining and contrasting conventional and tail-biting trellises. We point out that both types of trellises may be defined as graph-theoretic objects (cf. [61]) or, equivalently, as state-space realizations of a system (code). In this paper, we primarily use the latter viewpoint.

Conventional Trellises: A *conventional trellis* T is defined on an ordered *time axis*, which is usually a subset of the integers, namely, $I \subseteq \mathbb{Z}$. For a block code, the time axis I is finite and may be taken as $I = \{0, 1, \cdots, m\}$, where m is the length of the code in trellis sections.

A finite set S_j, called a *state space*, is defined for each time $j \in I$, with $|S_0| = |S_m| = 1$ by convention. Further, an output (or label) alphabet A_j is defined for each $j = 0, 1, \cdots, m-1$. For each such j, the *trellis section* connecting times j and $j+1$ is defined as a subset $T_j \subseteq S_j \times A_j \times S_{j+1}$ that specifies the allowed combinations (s_j, a_j, s_{j+1}) of states $s_j \in S_j$, output symbols $a_j \in A_j$, and states $s_{j+1} \in S_{j+1}$. Such allowed combinations are called *trellis branches*, or edges. A *trellis path* $(\boldsymbol{s}, \boldsymbol{a}) \in T$ is a state/output sequence pair such that the state sequence $\boldsymbol{s}$ is in the state-sequence space $\mathcal{S} = S_0 \times S_1 \times \cdots \times S_m$, the output sequence $\boldsymbol{a}$ is in the output sequence-space $\mathcal{A} = A_0 \times A_1 \times \cdots \times A_{m-1}$, and for

each $j = 0, 1, \cdots, m-1$, the triple (s_j, a_j, s_{j+1}) is a branch in T_j. The code represented by T is the subset $\mathbb{C}(T) \subseteq \mathcal{A}$ of all output sequences $\boldsymbol{a}$ corresponding to all possible trellis paths $(\boldsymbol{s}, \boldsymbol{a})$ in T.

A conventional trellis may be thought of as an edge-labeled directed graph $T = (\mathcal{S}, \mathcal{A}, E)$ in which the set of vertices $\mathcal{S}$ is the union of the sets $S_0, S_1, \cdots, S_m$, and the set of edges E consists of the trellis branches $(s_j, a_j, s_{j+1}) \in T_j$, so that each edge connects a vertex $s_j \in S_j$ to a vertex $s_{j+1} \in S_{j+1}$ and is labeled by $a_j \in A_j \subseteq \mathcal{A}$, for some j. The trellis may be also regarded as a representation of all possible state sequences and corresponding output sequences of a state-space realization of an encoder for $\mathbb{C}$, whose state space at time j corresponds to S_j for all $j \in I$.

Tail-Biting Trellises: In contrast, a *tail-biting trellis* T is defined on a circular time axis I of length m, which we will identify with $\mathbb{Z}_m = \{0, 1, \cdots, m-1\}$, the integers modulo m. All index arithmetic will be implicitly performed modulo m.

For each time $j \in \mathbb{Z}_m$, a finite state space S_j and a symbol alphabet A_j are defined. The tail-biting trellis section connecting times j and $j+1$ is again defined by an allowed branch subset $T_j \subseteq S_j \times A_j \times S_{j+1}$. In particular, the trellis section connecting times $j = m-1$ and $j+1 = 0$ is defined as a subset $T_{m-1} \subseteq S_{m-1} \times A_{m-1} \times S_0$. Again, a trellis path is a state/output sequence pair $(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}$ consisting of a sequence of branches that satisfy all the constraints, namely,

$$(\boldsymbol{s}, \boldsymbol{a}) \in T \Leftrightarrow (s_j, a_j, s_{j+1}) \in T_j, \quad \text{for all } j = 0, 1, \cdots, m-1.$$

The code represented by T is the subset $\mathbb{C}(T) \subseteq \mathcal{A}$ of all output sequences $\boldsymbol{a} \in \mathcal{A}$ corresponding to all possible trellis paths $(\boldsymbol{s}, \boldsymbol{a})$ in T.

Alternatively, a tail-biting trellis may be defined on a sequential time axis $I = \{0, 1, \cdots, m\}$, with the restrictions that $S_m = S_0$, and the valid paths $(\boldsymbol{s}, \boldsymbol{a}) \in T$ are those for which $s_m = s_0$. In other words, the code $\mathbb{C}(T)$ represented by T is the set of all output sequences $\boldsymbol{a} \in \mathcal{A}$ corresponding to those paths in T that start and end at the same state.

Remark: Notice that a conventional trellis may be conveniently regarded as a special case of a tail-biting trellis, with $|S_0| = |S_m| = 1$. Conversely, a tail-biting trellis that contains a state space of size 1 at some time $j \in I$ may be regarded as (a shifted version of) a conventional trellis.

We now proceed with some more definitions. The *state-complexity profile* of a tail-biting trellis is the ordered set $\{|S_0|, |S_1|, \cdots, |S_{m-1}|\}$. The *maximum state complexity* of a trellis is defined as $\mathcal{S}_{\max} = \max\{|S_0|, |S_1|, \cdots, |S_{m-1}|\}$. Generally, our goal is to minimize $\mathcal{S}_{\max}$ over all possible coordinate orderings and choices of generators. Such a tail-biting trellis will be called *μ-minimal*, or simply *minimal*. A secondary goal is to minimize the product of all state-space sizes, again over all coordinate permutations and choices of generators. Such a trellis will be called *π-minimal*. For a detailed treatment of various notions of minimality for tail-biting trellises, see [30]–[32].

A tail-biting trellis T will be called *cyclic* if all trellis sections T_j are equal; then all state spaces are equal, all output alphabets A_j are equal, and the code $\mathbb{C}(T)$ is necessarily cyclic on the time axis $\mathbb{Z}_m$. (If the alphabets A_j involve multiple symbols, then $\mathbb{C}(T)$ is usually called a *quasi-cyclic* code). A tail-biting trellis T will be called *structurally invariant* if all state spaces are equal and the set of allowed state transitions $(s_j, s_{j+1}) \in S_j \times S_{j+1}$ corresponding to the branches $(s_j, a_j, s_{j+1}) \in T_j$ is the same for all j. The distinction between a structurally invariant trellis T and a cyclic trellis T^* is that the *labels* of the branches may be time-varying in T, but not in T^*.

B. Specification of Tail-Biting Trellises by Generator Matrices

There is a well-developed theory [22], [34], [50] that, given a fixed coordinate ordering, specifies an essentially unique minimal conventional trellis for a linear or group code $\mathbb{C}$. No similar theory exists for the construction of minimal tail-biting trellises. The fundamental reason is that the time axis I is not ordered, so a "cut" of I does not divide it into "past" and "future." Development of a theory of minimal realizations for systems defined on unordered time axes is an important but difficult open problem in system theory.

Nonetheless, we now show that a generator matrix for a linear code $\mathbb{C}$, or a set of generators for a group code $\mathbb{C}$, may be used to specify a tail-biting trellis for $\mathbb{C}$. For all linear codes considered in this paper, we will be able to prove that the resulting tail-biting trellises are minimal.

By *minimal* we mean that the maximum number of states $\mathcal{S}_{\max}$ is minimized, and furthermore, that subject to the minimization of $\mathcal{S}_{\max}$ the product of all state-space sizes is minimized, among all tail-biting trellis representations for the corresponding code, under all possible coordinate permutations. Notice that this minimality requirement is somewhat weaker than the well-known [22], [34], [50] definition of minimality for conventional trellises, which requires simultaneous minimization of the number of states $|S_j|$ at *each* time $j \in I$. The latter definition is evidently not appropriate in the tail-biting context. As we shall see, one can always construct a tail-biting trellis, for any linear or group code $\mathbb{C}$, which has a *single* state at any prescribed time $j \in I$. On the other hand, our notion of minimality is also much stronger than the conventional definition, since it includes minimization over all possible permutations of the time axis, whereas the conventional definition pertains only to a fixed time axis. We should point out, however, that it is not clear whether this is the most appropriate definition of minimality for tail-biting trellises.

Let $\mathbb{C}$ be a linear or group block code of length m, and let all symbol alphabets A_j be equal to a common alphabet $\mathcal{A}$, so that the output sequence space is $\mathcal{A}^m$. In the linear case, $\mathcal{A}$ is a vector space over a field $\mathbb{F}$, and $\mathbb{C}$ is a subspace of the direct-product vector space $\mathcal{A}^m$. In the group case, $\mathcal{A}$ is a group and $\mathbb{C}$ is a subgroup of the direct-product group $\mathcal{A}^m$. Notice that every linear code is also a group code (over the additive group of $\mathbb{F}$), but not vice versa.

We also need to define *linear trellises* and *group trellises*. To this end, we follow the approach of [30] and consider the code $\mathcal{S}(T)$ that consists of all the trellis paths $(\boldsymbol{s}, \boldsymbol{a}) \in T$. Thus $\mathcal{S}(T)$ is a subset of $\mathcal{S} \times \mathcal{A}$. (In the factor-graph terminology, $\mathcal{S}(T)$ is simply the code that results by making all sites visible.)

If all the state spaces $S_0, S_1, \cdots, S_m$ and all the symbol alphabets $A_0, A_1, \cdots, A_{m-1}$ are vector spaces over a field, then $\mathcal{S} = S_0 \times S_1 \times \cdots \times S_m$ and $\mathcal{A} = A_0 \times A_1 \times \cdots \times A_{m-1}$ are the corresponding direct-product vector spaces. Similarly, if all the state spaces and symbol alphabets are groups, then $\mathcal{S}$ and $\mathcal{A}$ are the corresponding direct-product groups. We say that T is a *linear trellis* if $\mathcal{S} \times \mathcal{A}$ is a vector space and the code $\mathcal{S}(T)$ is a subspace of $\mathcal{S} \times \mathcal{A}$. We say that T is a *group trellis* if $\mathcal{S} \times \mathcal{A}$ is a group and $\mathcal{S}(T)$ is a subgroup of $\mathcal{S} \times \mathcal{A}$. Note that these definitions apply to both conventional and tail-biting trellises. Also notice that a linear (group) code $\mathbb{C}$ can be always represented by both linear (group) and nonlinear (nongroup) trellises.

Remark: If a trellis T is treated as a graph-theoretic object $T = (\mathcal{S}, \mathcal{A}, E)$, then the state spaces of T are, by definition, just sets of vertices without any algebraic structure. In this case, we say that a trellis T is a linear (group) trellis if *there exists* a labeling of the vertices of T such that $\mathcal{S}(T)$ becomes a linear (group) code. For much more on linear trellises see [30] and [31], where it is shown that trellis linearity is essentially a graph-theoretic property. It is also shown in [31] how to decide whether a given (unlabeled) trellis T is a linear (group) trellis.

Now let $\boldsymbol{G} = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_k\} \subset \mathcal{A}^m$ be a set of k generators that generate a code $\mathbb{C}$ under the appropriate definition of "generate" for the linear or group cases, respectively. In other words,

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

where $\langle \boldsymbol{g}_i \rangle$ is the one-dimensional vector space generated by $\boldsymbol{g}_i$ in the linear case, or the cyclic group generated by $\boldsymbol{g}_i$ in the group case. In the linear case, $\boldsymbol{G}$ may be regarded as a $k \times n$ generator matrix for $\mathbb{C}$, and $\mathbb{C}$ is the set of all linear combinations $\boldsymbol{uG} = \sum_i u_i \boldsymbol{g}_i$ as $\boldsymbol{u}$ ranges over $\mathbb{F}^k$.

For simplicity, we will describe the trellis construction in detail only for the linear case. The group case is similar, as we will show later in this section by example.

Conventional Trellises: An encoder for $\mathbb{C}$ may be obtained by realizing each generator $\boldsymbol{g}_i$ independently, as follows. The *span* of a generator is defined as the interval $[j, j']$ from its first nonzero component g_{ij} to its last nonzero component $g_{ij'}$, where the terms "first" and "last" are defined as usual by the ordering of the time axis $I \subseteq \mathbb{Z}$. Thus we may assume that $j' \geq j$. Realization of $\boldsymbol{g}_i$ involves an input $u_i \in \mathbb{F}$ at time j, a memory that stores u_i during the interval $(j, j']$ of length $j' - j$, and logic that produces the output $u_i \boldsymbol{g}_i$ during $[j, j']$. We say that $\boldsymbol{g}_i$ is *active* during the interval $(j, j']$, and call $(j, j']$ the *active interval* of $\boldsymbol{g}_i$.

Fig. 1 illustrates such a realization with a trellis diagram, for the generator $\boldsymbol{g}_i = (0, 1, 0, 2, 0, 0)$ over the ternary field $\mathbb{F}_3$ (or equivalently, the group $\mathbb{Z}_3$). The generator span is $[1, 3]$. A ternary state space is thus needed during the active interval $(1, 3] = \{2, 3\}$, whose length is 2; at other times only a trivial state space of size 1 is needed.

An encoder for $\mathbb{C}$ is then realized simply by summing the output sequences of each generator realization, which yields $\boldsymbol{uG} = \sum u_i \boldsymbol{g}_i$. The corresponding trellis may be thought of as the product of the elementary trellises of each generator $\boldsymbol{g}_i$. Its

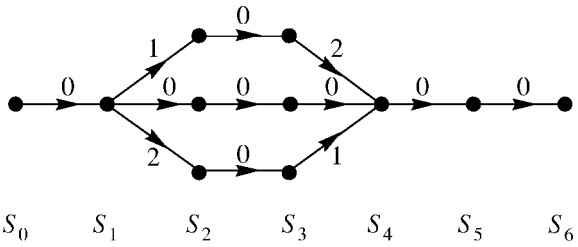


Fig. 1. Trellis diagram of a realization of a generator $\boldsymbol{g}_i = (0, 1, 0, 2, 0, 0)$ over $\mathbb{F}_3$.

state space at any time j is thus a vector space $S_j = \mathbb{F}^{\delta_j}$ whose dimension δ_j is the number of generators that are active at time j. For a more precise definition of the *product* of trellises, see [34], [30], and [31].

It has been shown in [17], [34], and [46] that if $\mathbb{C}$ is a linear code and $\boldsymbol{G}$ is in a *trellis-oriented* or *minimal-span* form, then this construction results in a linear trellis which is the unique minimal trellis for $\mathbb{C}$. Moreover, an arbitrary generator matrix can be reduced to a minimal-span form by a straightforward greedy sequence of row operations [34]. (For a group code, this generator-based construction produces a group trellis, although the minimal group trellis may not be realizable in this form [22]. Nonetheless, this construction will suffice for our purposes.)

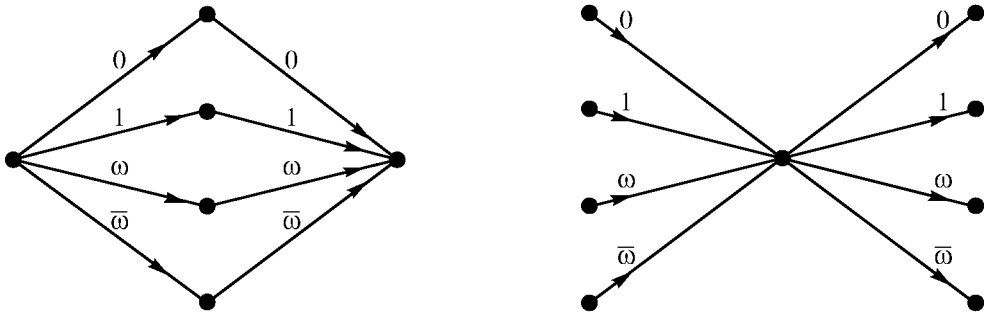
Tail-Biting Trellises: A tail-biting trellis for $\mathbb{C}$ can be constructed from a generator matrix $\boldsymbol{G}$ for $\mathbb{C}$ in a similar manner. However, in view of the circular time axis $I = \mathbb{Z}_m$, the definitions of "interval," "first," "last," and "span" must be revisited.

Since all arithmetic is modulo m, an interval on $\mathbb{Z}_m$ is defined in a circular ("wrap-around") fashion. For example, the interval $[5, 1] \subset \mathbb{Z}_8$ is $\{5, 6, 7, 0, 1\}$. Thus it is quite possible to have an interval $[j, j']$ with $j' < j$. It follows that the "first" nonzero component of a generator $\boldsymbol{g}_i$ may be chosen as any nonzero component of $\boldsymbol{g}_i$. That is, the beginning time j of the span of $\boldsymbol{g}_i$ is a *design parameter*. The index of the "last" nonzero component of $\boldsymbol{g}_i$ is then the least integer such that the span $[j, j']$ (modulo m) covers all of the nonzero components of $\boldsymbol{g}_i$. The active interval is again $(j, j']$. Its length is $j' - j$ if $j' \geq j$, and $m - j + j'$ otherwise.

Each generator is then realized independently, just as for conventional trellises. If the (circular) span of $\boldsymbol{g}_i$ is $[j, j']$, then a state space isomorphic to $\mathbb{F}$ is needed during the circular active interval $(j, j']$; the state is zero during the complementary interval $(j', j]$. The output sequence $\boldsymbol{uG} = \sum_i u_i \boldsymbol{g}_i$ is again obtained by summing the output sequences of these k state machines. The aggregate state space is given by $S_j = \mathbb{F}^{\delta_j}$ for all $j \in \mathbb{Z}_m$, where δ_j is the number of generators that are active at time j, according to *the span parameters that we have selected*.

The above discussion briefly describes what is known [30], [31], [34] as the *product construction* of trellises. Recently, Kötter and Vardy [30], [31] have shown that *any* linear trellis, either conventional or tail-biting, for a linear code $\mathbb{C}$ may be constructed in this way.

Example 1: Let $\mathbb{C}$ be the $(2, 1, 2)$ repetition code over the quaternary field $\mathbb{F}_4 = \{0, 1, \omega, \overline{\omega}\}$, where $\overline{\omega} = \omega^2 = \omega + 1$.



(a) (b)

Fig. 2. Conventional and tail-biting linear trellises for the $(2,1,2)$ repetition code over $\mathbb{F}_4$.

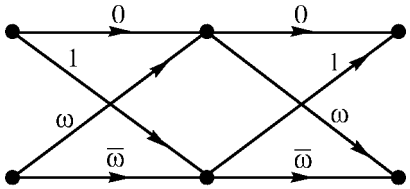


Fig. 3. Group tail-biting trellis for the $(2,1,2)$ repetition code.

That is, $\mathbb{C} = \{00, 11, \omega\omega, \overline{\omega}\,\overline{\omega}\}$. This is a cyclic linear code over $\mathbb{F}_4$; it is also a group code over the additive group of $\mathbb{F}_4$. We use this simple code to illustrate not only the tail-biting trellis construction above, but also the following points.

Freedom of choice: A tail-biting trellis depends on the choice of spans;

Linear versus group trellis: Remarkably, a group tail-biting trellis for $\mathbb{C}$ may be simpler than any linear tail-biting trellis for $\mathbb{C}$.

As a linear code, $\mathbb{C}$ has a single generator $\boldsymbol{g} = (11)$. If we choose the span of $\boldsymbol{g} = (11)$ as $[0,1]$, then s_0 is zero and s_1 is an element of $\mathbb{F}_4$. Hence we get a conventional linear trellis with $|S_0| = 1$, $|S_1| = 4$, and $|S_2| = |S_0| = 1$, which is illustrated in Fig. 2(a). However, we could equally well say that the span of $\boldsymbol{g} = (11)$ is the wrap-around interval $[1,0]$. Then we obtain a tail-biting linear trellis with $|S_0| = 4$, $|S_1| = 1$, and $S_2 = S_0$, which is shown in Fig. 2(b).

Now let us consider $\mathbb{C}$ as a group code over the additive group of $\mathbb{F}_4$, denoted by $V = \{0, 1, \omega, \overline{\omega}\}$, which is isomorphic to $\mathbb{Z}_2^2$. As a group code, $\mathbb{C}$ is generated by the two generators $\boldsymbol{g}_1 = (11)$ and $\boldsymbol{g}_2 = (\omega\omega)$, each of which generates a two-word subcode isomorphic to $\mathbb{Z}_2$. The code $\mathbb{C}$ is the direct sum of these two subcodes. Now, if we say that the span of both generators is $[0,1]$, then each subcode has state-complexity profile $\{1,2\}$, and their product yields the trellis of Fig. 2(a) again. If we say that the span of both generators is $[1,0]$, then we get the trellis of Fig. 2(b) again. However, if we say that the spans of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are $[0,1]$ and $[1,0]$, respectively, then we obtain the group tail-biting trellis shown in Fig. 3, in which $|S_0| = |S_1| = 2$.

Thus the maximum state-space size $\mathcal{S}_{\max}$ of a group tail-biting trellis for $\mathbb{C}$ can be less than that of any linear tail-biting trellis for $\mathbb{C}$. This is quite unexpected, because this cannot occur for conventional trellises. The difference arises because there is no freedom to choose generator spans in the construction of conventional trellises for linear or group codes [22].

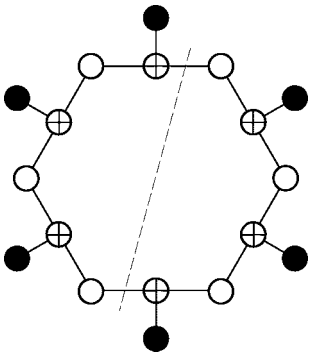


Fig. 4. Generic factor graph for a tail-biting trellis of length $m = 6$, with edge cut set.

For this simple code, it is not clear that the tail-biting trellis of Fig. 3, which minimizes $\mathcal{S}_{\max}$, is really any simpler than the minimal conventional trellis of Fig. 2(a). While the tail-biting trellis has fewer states, both trellises have a total of eight branches. For more complex codes, such as the $(24, 12, 8)$ Golay code, it will be easier to argue that the tail-biting trellis is genuinely simpler.

Notice that none of these trellises is cyclic, and moreover $\mathbb{C}$ clearly has no tail-biting trellis that is both cyclic and minimal, because any 2-state tail-biting trellis with identical sections cannot generate the $(2,1,2)$ repetition code. We note that the group code $\mathbb{C}' = \{00, 1\omega, \omega 1, \overline{\omega}\,\overline{\omega}\}$ generated by $\boldsymbol{g}_1 = (1\omega)$ and $\boldsymbol{g}_2 = (\omega 1)$ is an alternative cyclic $(2,1,2)$ group code over V that has a minimal 2-state cyclic tail-biting trellis. However, $\mathbb{C}'$ is not a linear code over $\mathbb{F}_4$.

C. Factor Graphs and the Square Root Bound

A tail-biting trellis may be also thought of as a *factor graph*, as defined in [36], [20], [67], and [68]. A factor graph for a tail-biting trellis with $m = 6$ sections is illustrated in Fig. 4. In a factor graph, there are three kinds of nodes, namely,

symbol nodes: representing symbol alphabets A_j (filled circles);

state nodes: representing state spaces S_j (open circles);

check nodes: representing constraints T_j on the values of adjacent symbol and state nodes (crossed circles).

For a conventional or tail-biting trellis, there is a check node corresponding to each trellis section, indicating the allowed subset $T_j \subseteq S_j \times A_j \times S_{j+1}$, namely, the allowable state transitions and their associated output symbols. Check nodes are connected by edges only to symbol/state nodes, and vice versa. A factor graph is thus a bipartite graph, where symbol and state nodes are regarded as being of the same type. Moreover, each edge in the factor graph is regarded as having the same alphabet as its associated symbol or state node.

A cut set in a graph is a set of edges whose removal divides the graph into two disconnected graphs. The *cut-set bound* is a general lower bound that permits the application of known lower bounds on the minimal state-space sizes of conventional trellises to the product of the alphabet sizes of the edges in any cut set in an arbitrary factor graph. Wiberg, Loeliger, and Kötter [68], [67] state and prove the cut-set lower bound for linear codes and graphs, but the principles involved are completely general. Because most of our minimality results are based on this bound, we state and prove it here in full generality.

Theorem 1 (Cut-Set Lower Bound): Let I be the symbol index set of a code $\mathbb{C}$, let $\mathcal{S}$ be an edge cut set in a factor graph for $\mathbb{C}$, and let J and $I-J$ be the subsets of symbol indices that index the symbol nodes in the two disconnected parts of the graph created by the removal of $\mathcal{S}$. Then the product of the sizes of the state or symbol alphabets associated with the edges in $\mathcal{S}$ is at least as large as the minimal state-space size in a conventional trellis for $\mathbb{C}$, given a coordinate ordering and a past/future cut such that the symbol nodes indexed by J are in the past and those indexed by $I-J$ are in the future.

Proof: We will show that the Cartesian product

$$X = \prod_{s \in \mathcal{S}} X_s$$

of the state or symbol alphabets $\{X_s : s \in \mathcal{S}\}$ associated with the edges in $\mathcal{S}$ can serve as the state space in the center of a conventional two-section trellis for $\mathbb{C}$. We call an element $\boldsymbol{x} = (x_s, s \in \mathcal{S})$ of X an *$\mathcal{S}$-configuration*, and a set of possible values $(\boldsymbol{s}, \boldsymbol{a})$ for all state and symbol nodes in the factor graph for $\mathbb{C}$ a *graph configuration*. We say that a graph configuration $(\boldsymbol{s}, \boldsymbol{a})$ is consistent with an $\mathcal{S}$-configuration $\boldsymbol{x}$ if its restriction to $\mathcal{S}$ is $\boldsymbol{x}$. Now let $P_J(\mathbb{C}|\boldsymbol{x})$ denote the set (possibly empty) of projections of valid graph configurations that are consistent with $\boldsymbol{x}$ onto the symbol index subset J and define $P_{I-J}(\mathbb{C}|\boldsymbol{x})$ similarly. The set of all codewords in $\mathbb{C}$ that are consistent with $\boldsymbol{x}$ is then the concatenation $P_J(\mathbb{C}|\boldsymbol{x}) \times P_{I-J}(\mathbb{C}|\boldsymbol{x})$, since the two disconnected parts of the factor graph created by the removal of $\mathcal{S}$ are independent. Furthermore, $\mathbb{C}$ is the union of all such concatenations

$$\mathbb{C} = \bigcup_{\boldsymbol{x} \in X} P_J(\mathbb{C}|\boldsymbol{x}) \times P_{I-J}(\mathbb{C}|\boldsymbol{x}).$$

Consequently, there exists a two-section conventional trellis for $\mathbb{C}$ with state space X such that the set of branches from the initial node to the state $\boldsymbol{x} \in X$ corresponds to the set $P_J(\mathbb{C}|\boldsymbol{x})$, and the set of branches to the final node from the state $\boldsymbol{x}$ corresponds to the set $P_{I-J}(\mathbb{C}|\boldsymbol{x})$. □

Remark: Note that if either $P_J(\mathbb{C}|\boldsymbol{x})$ or $P_{I-J}(\mathbb{C}|\boldsymbol{x})$ is empty for a state $\boldsymbol{x} \in X$, then this state may be discarded. The cut-set lower bound may thus be tightened by removing all such states.

For a tail-biting trellis, a minimal cut set consists of two edges whose alphabets correspond to any two distinct state spaces S_j and $S_{j'}$. The cut-set bound thus lower-bounds the product $|S_j||S_{j'}|$ of state-space sizes for any index pair $\{j, j'\}$. For our purposes, it will suffice to consider only diametrically opposite pairs $\{S_j, S_{j+m/2}\}$, as illustrated in Fig. 4. This leads to the following lower bound, which again is essentially due to Wiberg, Loeliger, and Kötter [68], [67].

Corollary 2 (Square-Root Lower Bound): Let $\mathbb{C}$ be a code of even length m, and let $\mathcal{S}_{\text{mid}}$ be the minimum possible state-space size of a conventional trellis for $\mathbb{C}$ at its midpoint, under any coordinate ordering. Then

$$\mathcal{S}_{\max} \geq \sqrt{\mathcal{S}_{\text{mid}}}$$

where $\mathcal{S}_{\max}$ is the maximum state complexity of any tail-biting trellis for $\mathbb{C}$. Equality holds if and only if $|S_j| = \sqrt{\mathcal{S}_{\text{mid}}}$ for all $j \in \mathbb{Z}_m$.

Proof: Choose a cut set consisting of two diametrically opposite edges with alphabets S_j, $S_{j+m/2}$. Then

$$|S_j|\,|S_{j+m/2}| \geq \mathcal{S}_{\text{mid}}$$

by Theorem 1. Thus either

$$|S_j| \geq \sqrt{\mathcal{S}_{\text{mid}}}$$

or

$$|S_{j+m/2}| \geq \sqrt{\mathcal{S}_{\text{mid}}}$$

with equality if and only if

$$|S_j| = |S_{j+m/2}| = \sqrt{\mathcal{S}_{\text{mid}}}.$$

Since j is arbitrary, the corollary follows. □

Clearly, a tail-biting trellis is minimal if the square-root lower bound is met with equality. In this case, we necessarily have

$$|S_j| = \sqrt{\mathcal{S}_{\text{mid}}}, \qquad \text{for all } j \in \mathbb{Z}_m$$

in view of Corollary 2. It is easy to see that such a trellis simultaneously minimizes the maximum number of states, the total number of states, and the product of all state-space sizes.

For example, the minimum possible state-space size at the midpoint of a conventional trellis for the $(2, 1, 2)$ repetition code $\mathbb{C}$ over $\mathbb{F}_4$ is $\mathcal{S}_{\text{mid}} = 4$ as in Fig. 2(a). From Corollary 2, it follows that $\mathcal{S}_{\max} \geq 2$ in a tail-biting trellis. Fig. 3 shows that there exists a minimal tail-biting trellis for $\mathbb{C}$ with state-complexity profile $\{2, 2\}$. Notice that the state-complexity profiles $\{1, 4\}$ and $\{4, 1\}$ of the two trellises in Fig. 2 both satisfy the cut-set bound $|S_j||S_{j+1}| \geq \mathcal{S}_{\text{mid}}$ with equality, at all positions j. However, they do not meet the square-root lower bound of Corollary 2.

III. The Golay Code

It is well known [17], [50] that the minimal possible number of states at the midpoint of any conventional trellis for the $(24, 12, 8)$ binary Golay code is $\mathcal{S}_{\text{mid}} = 256$. The square-root

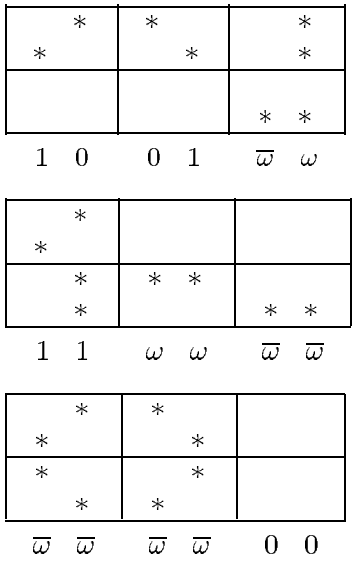
lower bound therefore implies that a tail-biting trellis for the Golay code must have at least $\mathcal{S}_{\max} = 16$ states, and that this is achievable if and only if *every* state-space size is equal to 16.

In this section we exhibit such a minimal tail-biting trellis for the Golay code. We observe that by "unwrapping" the generators used to construct this trellis, we obtain a "Golay convolutional code" with nice dynamical and algebraic structure and a good performance/complexity tradeoff.

A. Minimal Tail-Biting Trellis

Our objective is to find a generator matrix for the Golay code which has the structure of (1). Our key tool for finding an appropriate coordinate permutation is the Miracle Octad Generator (MOG) representation of the Golay code, which we now briefly describe.

The MOG is a 4×6 array. We specify binary vectors of length 24 by indicating their nonzero positions in this array. For example, the three vectors displayed in the MOG notation below



(2)

are codewords of $\mathbb{C}_{24}$. In general, the map φ: $\mathbb{F}_2^4 \to \mathbb{F}_4$ defined by $(abcd) \to a \cdot 0 + b \cdot 1 + c \cdot \omega + d \cdot \overline{\omega}$ can be applied to the six columns of the MOG array to produce a composite *projection* map from $\mathbb{F}_2^{24}$ to $\mathbb{F}_4^6$. With this notation, the Golay code $\mathbb{C}_{24}$ may be defined as the set of all the 4×6 binary arrays such that the parity of each column is equal to the parity of the top row and the projection of the array is a codeword of the $(6, 3, 4)$ hexacode H_6 over $\mathbb{F}_4$. For more details on the hexacode, see the next section. For more details on the foregoing definition of $\mathbb{C}_{24}$, and the many remarkable properties of the MOG, see Conway and Sloane [11, Ch. 11].

Here, we seek labelings of the MOG coordinates that produce a generator matrix of the form (1) for $\mathbb{C}_{24}$. One such coordinate labeling is as follows:

11 3	1 17	12 18
2 9	8 22	16 19
6 4	0 23	13 15
10 5	7 14	20 21

(3)

Using the definition of the Golay code in terms of the MOG, it is easy to verify that for each $i = 0, 2, 4, \cdots, 22$, the positions labeled $i, i+1, \cdots, i+9$ (modulo 24) support a Golay codeword. For example, three such codewords corresponding to $i = 18$, $i = 20$, and $i = 22$ are shown in (2).

For a discussion of how the labeling in (3) was found, we refer the reader to [9]. There, we also show that this labeling is essentially unique, up to automorphisms of $\mathbb{C}_{24}$.

Permuting the twelve codewords corresponding to $i = 0, 2, 4, \cdots, 22$, according to the labeling (3) produces the generator matrix for the Golay code (see (4) at the bottom of this page). It is easy to see that $\boldsymbol{G}_{24}$ conforms to the structure of (1). One can also verify that $\boldsymbol{G}_{24}$ has rank 12. Hence it generates the Golay code.

This can be also deduced directly, without relying on the MOG coordinates, as follows. First notice that all the generators in (4) are orthogonal to each other. Hence the code generated by $\boldsymbol{G}_{24}$ is self-dual. Together with the fact that all the generators in (4) have Hamming weight 8, this implies that the weight of all the codewords in the code generated by $\boldsymbol{G}_{24}$ is a multiple of 4 (see Lemma 3, below). It is easy to verify (for instance, using the trellis below) that this code has no words of weight 4. Hence $\boldsymbol{G}_{24}$ generates a $(24, 12, 8)$ linear code. On the other hand, it is well known [51] that the Golay code is the unique binary code with these parameters.

We now construct a tail-biting trellis for $\mathbb{C}_{24}$ based on the generator matrix in (4). Let the symbol alphabet A be $\mathbb{F}_2^2$, the set of binary 2-tuples, and let the time axis be identified with $\mathbb{Z}_{12}$. That is, we propose to construct a tail-biting trellis

$$\boldsymbol{G}_{24} = \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \boldsymbol{g}_3 \\ \boldsymbol{g}_4 \\ \boldsymbol{g}_5 \\ \boldsymbol{g}_6 \\ \boldsymbol{g}_7 \\ \boldsymbol{g}_8 \\ \boldsymbol{g}_9 \\ \boldsymbol{g}_{10} \\ \boldsymbol{g}_{11} \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} 1&1&0&1&1&1&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&1&1&1&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&1&1&1&0&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&1&1&1 \\ 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&1 \\ 0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0 \\ 1&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\ 0&1&1&1&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \end{bmatrix}. \quad (4)$$

with $m = 12$ sections. For each generator in (4), we choose the natural circular span of length 5. Namely, the span of $\boldsymbol{g}_i$ is defined as the circular interval $[i, i+4]$, for all $i = 0, 1, \cdots, 11$. A state-space realization of $\boldsymbol{g}_i$ thus involves a binary state space that is active during the interval $(i, i+4]$ of length 4. It follows that there are precisely four binary state spaces active at each time j. Hence the state-space size $|S_j|$ is equal to $2^4 = 16$ for all $j \in \mathbb{Z}_{12}$. The resulting trellis is minimal, by the square-root lower bound.

The generator matrix in (4) and the corresponding tail-biting trellis are not cyclic. (We will show in Section III-B that a cyclic 16-state generator matrix for $\mathbb{C}_{24}$ does not exist.) The generator matrix $\boldsymbol{G}_{24}$ does have period 4, however. That is, the corresponding three-section tail-biting trellis over the symbol alphabet $\mathbb{F}_2^8$ is cyclic. It is also reversible.

Fig. 5 shows four sections of the 16-state tail-biting trellis for the Golay code resulting from our construction. The complete trellis is just the three-fold repetition of Fig. 5. Furthermore, the first and fourth sections in Fig. 5 are identical. Notice that

- there is a path from each state in S_j to each state in S_{j+4}—in all, 256 paths;
- each 8-tuple in $\mathbb{F}_2^8$ occurs as the output sequence for one and only one of the 256 paths.

The second property follows from the fact that there is only one all-zero path, combined with the group structure of $\mathbb{C}_{24}$, or alternatively from the fact that every eight consecutive columns in $\boldsymbol{G}_{24}$ are linearly independent. Both properties also follow directly from examination of Fig. 5.

In the language of system theory, the first property implies that the trellis is 4-controllable (any state can be reached from any other in four time units), while the second property implies that the trellis is 4-observable (observation of four time units of the output sequence suffices to determine the state sequence during those times).

Furthermore, it follows from the structure of the generator matrix in (4) that $\mathbb{C}_{24}$ has a memory-4 shift-register (controller-form) encoder, with time-varying taps $\boldsymbol{g}_{j,j}$, $\boldsymbol{g}_{j,j-1}$, $\boldsymbol{g}_{j,j-2}$, $\boldsymbol{g}_{j,j-3}$, $\boldsymbol{g}_{j,j-4}$, driven by an input sequence $\boldsymbol{u} \in \mathbb{F}_2^{12}$. This encoder is depicted in Fig. 6.

Notice that the tail-biting trellis of Fig. 5 is not cyclic, but is structurally invariant. Correspondingly, the encoder of Fig. 6 is not (cyclically) time-invariant, but its memory structure is time-invariant. In view of this memory structure, a Golay codeword is completely determined by any state 3-tuple of the form $(s_j, s_{j+4}, s_{j+8}) \in S_j \times S_{j+4} \times S_{j+8}$, where each of the three states may be freely chosen. Indeed, the number of possible choices of (s_j, s_{j+4}, s_{j+8}) is $16^3 = |\mathbb{C}_{24}|$.

Finally, we note that we have chosen the generator matrix $\boldsymbol{G}_{24}$ to be as symmetric as possible; in particular, to have period 4. However, it does not exhibit many well-known symmetries of the Golay code. For example, since the all-one codeword is generated by the input sequence $\boldsymbol{u} = 011001100110$, neither $\boldsymbol{G}_{24}$ nor the corresponding trellis of Fig. 5 makes it clear that $\mathbb{C}_{24}$ is self-complementary (contains the all-one codeword).

Remark: There is a standard construction for the Leech lattice Λ_{24} that is based on the Golay code. In Appendix A, this construction is used to obtain a 12-section tail-biting trellis for Λ_{24} with state-complexity profile $\{32, 64, \cdots, 64, 32, 64, \cdots, 64\}$.

B. The Golay Convolutional Code

The tail-biting idea has most commonly been used (cf. [44]) to obtain block codes from convolutional codes without incurring the rate loss that results from terminating a convolutional code with a "tail." Herein, we go in the opposite direction, and show that "unwrapping" a tail-biting representation of a good block code such as $\mathbb{C}_{24}$ can produce a good convolutional code.

Specifically, the first four rows of the matrix $\boldsymbol{G}_{24}$ in (4) may be used as generators for a 16-state rate-$1/2$ periodically time-varying binary linear convolutional code $\mathbb{C}_{\mathcal{G}}$, with period 4 time units (8 bits). We call $\mathbb{C}_{\mathcal{G}}$ the *Golay convolutional code*, because it is a natural counterpart of the Golay block code. Explicitly, $\mathbb{C}_{\mathcal{G}}$ is generated by

$$\begin{Bmatrix} 11 & 01 & 11 & 01 & 11 & 00 & 00 & 00 \\ 00 & 11 & 11 & 10 & 01 & 11 & 00 & 00 \\ 00 & 00 & 11 & 01 & 10 & 11 & 11 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 01 & 11 \end{Bmatrix}. \tag{5}$$

The 16-state rate-$1/2$ time-varying encoder of Fig. 6 may be used as an encoder for $\mathbb{C}_{\mathcal{G}}$, and the trellis of $\mathbb{C}_{\mathcal{G}}$ is the indefinite repetition of Fig. 5. Note, incidentally, that the set of generators in (5) is reversible, and thus so is the Golay convolutional code $\mathbb{C}_{\mathcal{G}}$.

The Golay convolutional code may be regarded either as a periodically time-varying rate-$1/2$ code with period 4, or as a time-invariant rate-$4/8$ "unit-memory" code. The former viewpoint leads to significantly less encoding and decoding complexity.

The Golay Convolutional Code is Type II: A linear code is said to be *even* if all codeword weights are even, and *doubly-even* if all codeword weights are multiples of 4. A linear code is said to be *self-orthogonal* if it is contained in its dual, and *self-dual* if it is equal to its dual.

Doubly-even self-dual (Type II) binary linear *block* codes have received much attention in algebraic coding theory (cf. [45, Ch. 19]). The Golay convolutional code $\mathbb{C}_{\mathcal{G}}$ is the first example known to us of a doubly-even self-dual (Type II) binary linear *convolutional* code. (As discussed below, it has turned up previously in exhaustive searches [42], [6], but without recognition of its Type II property or its connection to the Golay block code.) Its self-duality follows from inspection of its generators and their shifts. Its doubly-even property follows from Lemma 3 below, which is the counterpart of a well-known lemma [45, Problem 1.38] for binary linear block codes.

Definition: A set of generators for a convolutional code is said to be *noncatastrophic* if every finite code sequence can be expressed as the sum of a finite number of shifts of generators.

For example, the generator $(11\ 11)$ for a 2-state, rate-$1/2$, binary linear convolutional code $\mathbb{C}$ is catastrophic, since the

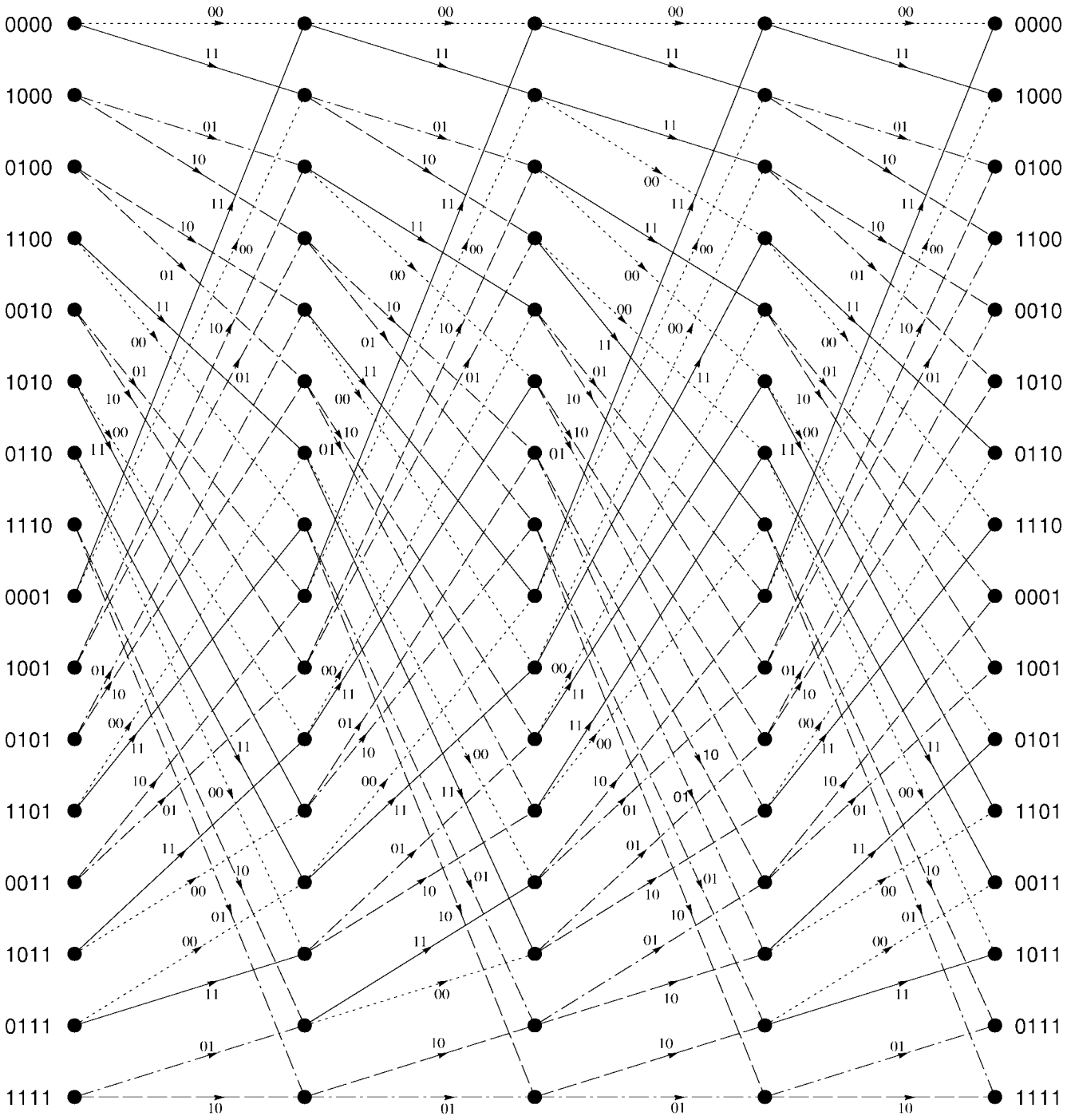


Fig. 5. Four sections of the minimal tail-biting trellis for the Golay code.

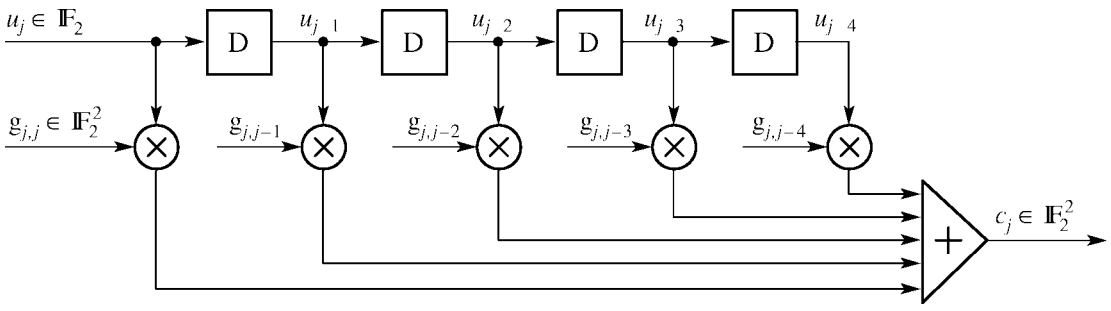


Fig. 6. Controller-form encoder for the Golay code.

finite code sequence $(11\ 00\ 00\ \cdots) \in \mathbb{C}$ is obtained from the infinite sum

$$(11\ 11\ 00\cdots)+(00\ 11\ 11\ 00\cdots)+(00\ 00\ 11\ 11\ 00\cdots)+\cdots.$$

This example shows why the noncatastrophic condition is needed in the following lemma. (This is also a good example for the proof of Lemma 4 below.)

Lemma 3: A block or convolutional binary linear code is doubly-even if and only if it is self-orthogonal and has a set of noncatastrophic doubly-even generators.

Proof: For any two binary sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, we have $\mathrm{wt}\,(\boldsymbol{x}+\boldsymbol{y}) = \mathrm{wt}\,(\boldsymbol{x})+\mathrm{wt}\,(\boldsymbol{y})-2\mathrm{wt}\,(\boldsymbol{x}*\boldsymbol{y})$, where $\boldsymbol{x}*\boldsymbol{y}$ stands for the componentwise product of the two sequences. Since the inner product $\langle \boldsymbol{xy} \rangle = 0$ if and only if $\mathrm{wt}\,(\boldsymbol{x}*\boldsymbol{y}) \equiv 0 \bmod 2$, it follows that $\mathrm{wt}\,(\boldsymbol{x}+\boldsymbol{y}) \equiv \mathrm{wt}\,(\boldsymbol{x}) + \mathrm{wt}\,(\boldsymbol{y}) \bmod 4$ if and only if $\langle \boldsymbol{xy} \rangle = 0$. The lemma now follows by finite induction if and only if the generators are noncatastrophic, since then every finite code sequence is a finite sum of generators. □

The generators of $\mathbb{C}_{\mathcal{G}}$ are doubly-even and self-orthogonal by inspection. Noncatastrophicity follows from the fact that there is no semi-infinite trellis path with an all-zero label sequence other than the zero-state path. This, in turn, follows from the 4-observability of this trellis, or by inspection of Fig. 5. Thus $\mathbb{C}_{\mathcal{G}}$ is doubly-even.

It is easy to check that the minimum (free) distance of $\mathbb{C}_{\mathcal{G}}$ is not 4; therefore, its minimum distance is $d = 8$. Moreover, its next distance is $d_{\mathrm{next}} = 12$. We observe that $\mathbb{C}_{\mathcal{G}}$ has a larger minimum distance than the maximum free distance of any time-invariant 16-state rate-$1/2$ code, which is known [12] to be $d = 7$. Riedel and Weiss [55] have shown that the performance improvement on an additive white Gaussian noise channel is about 0.2 dB.

It follows that a cyclic generator matrix for $\mathbb{C}_{24}$ over $\mathbb{F}_2^2$ of the form (1)—in other words, a 16-state quasi-cyclic representation of $\mathbb{C}_{24}$—does not exist. Otherwise, the first row $\boldsymbol{g}_0$ of such a generator matrix would then be the generator of a time-invariant 16-state rate-$1/2$ binary linear convolutional code with $d = 8$; but no such code exists.

The superiority of $\mathbb{C}_{\mathcal{G}}$ over any time-invariant code and the fact that Type II convolutional codes have not been previously observed are partially explained by the following lemma.

Lemma 4: A self-dual, doubly-even (Type II) binary linear *time-invariant* convolutional code of rate-$1/2$ does not exist.

Proof: A rate-$1/2$ time-invariant binary linear convolutional code $\mathbb{C}$ is generated by the time shifts of a single generator, which in conventional D-transform notation may be written as $\boldsymbol{g}(D) = [g_1(D), g_2(D)]$, where $g_1(D)$ and $g_2(D)$ are binary polynomials in the indeterminate D. (For example, in D-transform notation the generator $(11\ 11)$ is written as $[1+D,\ 1+D]$.) The generator $\boldsymbol{g}(D)$ is noncatastrophic if and only if $g_1(D)$ and $g_2(D)$ are relatively prime. The dual of $\mathbb{C}$, in the sense of the componentwise inner product used in Lemma 3, is the time-reverse of the dual of $\mathbb{C}$ as usually defined for convolutional codes, where orthogonality is based on sequence convolution. The usual dual is the convolutional code generated by $[g_2(D), g_1(D)]$. Therefore, the dual in the sense of Lemma 3 is generated by $[\overleftarrow{g_2}(D), \overleftarrow{g_1}(D)]$, where $\overleftarrow{g}(D)$ denotes the time-reverse of $g(D)$. Thus if $\mathbb{C}$ is self-dual, then $g_2(D) = \overleftarrow{g_1}(D)$ and $\boldsymbol{g}(D) = [g_1(D), \overleftarrow{g_1}(D)]$. Since $g_1(D)$ and $\overleftarrow{g_1}(D)$ have the same parity, it follows that $\boldsymbol{g}(D)$ is doubly-even if and only if $g_1(D)$ is even. But a binary polynomial is even if and only if it is divisible by $1 + D$. Thus if $g_1(D)$ and $g_2(D) = \overleftarrow{g_1}(D)$ are both even, then both are divisible by $1 + D$, so $\boldsymbol{g}(D)$ is catastrophic. Thus no noncatastrophic self-orthogonal doubly-even generator $\boldsymbol{g}(D)$ exists. □

Distance Structure of the Golay Convolutional Code: The number of minimum-weight code sequences of $\mathbb{C}_{\mathcal{G}}$ is easily enumerated with the aid of the trellis of Fig. 5. There are $N_8 = 49$ weight-8 sequences, which correspond to a total of 178 information bit errors. The shortest minimum-weight code sequences are the generators, and the longest minimum-weight code sequence is $(00110000\ 10001000\ 00100010\ 00001100)$. The latter sequence is produced by the encoder in Fig. 6 in response to the input sequence $\boldsymbol{u} = (0111\ 0110\ 1110)$.

Equivalent rate-$4/8$ codes have been previously discovered by Lee [42], and by Bocharova and Kudryashov [6]. In fact, this is the only doubly-even rate-$k/2k$ convolutional code to be found in the extensive tables of [6]. Bocharova and Kudryashov [7] have kindly provided the distance spectrum of $\mathbb{C}_{\mathcal{G}}$ as a rate-$4/8$ code in its four possible phases (for distances $8,\ 12,\ 16,\ \cdots$):

$$\begin{aligned}
&\text{Phase 1:} \quad 49, 1352, 38717, \cdots \\
&\text{Phase 2:} \quad 49, 1352, 38521, \cdots \\
&\text{Phase 3:} \quad 49, 1352, 38717, \cdots \\
&\text{Phase 4:} \quad 49, 1352, 38996, \cdots.
\end{aligned} \tag{6}$$

Gelblum [13] has computed the distance spectrum of $\mathbb{C}_{\mathcal{G}}$ as a time-varying rate-$1/2$ code in its four possible phases (for distances $8, 12, 16, 20, \cdots$)

$$\begin{aligned}
&\text{Phase 1:} \quad 10, 276, 7731, 216134, \cdots \\
&\text{Phase 2:} \quad 14, 388, 10833, 302964, \cdots \\
&\text{Phase 3:} \quad 15, 406, 11385, 318252, \cdots \\
&\text{Phase 4:} \quad 10, 282, 7872, 220154, \cdots.
\end{aligned} \tag{7}$$

Summing up the four phases in (7), we can compute the total distance spectrum of $\mathbb{C}_{\mathcal{G}}$ as a time-varying rate-$1/2$ code. This distance spectrum (per 8 code bits) is given by

$$\text{Total:} \quad 49, 1352, 37821, 1057504, \cdots.$$

This differs slightly from the rate-$4/8$ profiles computed in (6), for $d = 16, 20, \cdots$, which is not surprising. For example, a sequence of two weight-8 rate-$1/2$ code sequences that enter and leave the zero state within one 8-bit block can be counted as a single weight-16 code sequence in the rate-$4/8$ code. Finally, the average distance spectrum per 2 code bits is

$$\text{Average:} \quad 12.25, 338, 9455.25, 264376, \cdots.$$

This average distance spectrum should be used for comparison with the distance spectra of conventional rate-$1/2$ convolutional codes.

Comparison with Other Codes: As a rate-$4/8$ code, the Golay convolutional code may be compared with another, better known, family of rate-$4/8$ binary convolutional codes with $d = 8$ and only eight states. A code of this type was originally discovered by Lee [41] as a 16-state unit-memory code. Subsequently, Lauer [40] described an equivalent 8-state "partial-unit-memory" code. A general description of codes of the Lee–Lauer type is given in Appendix B, particularly to show their connection to the Turyn construction [17], [45] of the Golay block code.

A Lee–Lauer code has the same minimum distance $d = 8$ as the Golay convolutional code. It is even, but not self-dual and not doubly-even. The numbers of code sequences per four time units (8 bits) of weights $d = 8, 10, 12, 14, 16, \cdots$ are

$$29, 112, 616, 3248, 9968, \cdots$$

respectively. The multiplicity $N_8 = 29$ is about half that of $\mathbb{C}_{\mathcal{G}}$. However, a Lee–Lauer code has 112 weight-10 code sequences, whereas $\mathbb{C}_{\mathcal{G}}$ has none.

Although a Lee–Lauer code is nominally an 8-state code, its actual trellis complexity is somewhat greater than that of the 16-state code $\mathbb{C}_{\mathcal{G}}$ because it cannot be expressed as a time-varying rate-$1/2$ code with fewer than 64 states. As shown in [48], its state-complexity profile with 2-bit time units (sections) is $\{\cdots, 8, 32, 64, 32, 8, \cdots\}$ under the best possible coordinate permutation, as compared to $\{\cdots, 16, 16, 16, 16, 16, \cdots\}$ for $\mathbb{C}_{\mathcal{G}}$. The regular structure of the Golay convolutional code trellis is also an advantage. However, there has been considerable work on simplified decoding of Lee–Lauer codes; see [25], [28], [48], and references therein.

Overall, it seems fair to say that Lee–Lauer codes are slightly more complex and can achieve slightly better performance than $\mathbb{C}_{\mathcal{G}}$. Of course, both of these convolutional codes have a better performance/complexity tradeoff than the Golay code itself.

Finally, we note that, for block-code applications any tail-biting termination of $\mathbb{C}_{\mathcal{G}}$ produces an $(8b, 4b)$ linear self-dual doubly-even binary block code, characterized by the 16-state tail-biting trellis consisting of the b-fold repetition of Fig. 5. For $b \geq 3$, a minimum distance of $d = 8$ is achieved. The parameters of the first few codes in this sequence are $(8, 4, 4)$, $(16, 8, 4)$, $(24, 12, 8)$, $(32, 16, 8)$, $(40, 20, 8)$, and so forth [55]. The $(8, 4, 4)$, $(24, 12, 8)$, and $(32, 16, 8)$[1] codes have the largest possible minimum distance for their length and dimension. However, as the length grows, this sequence of codes obviously becomes asymptotically bad. On the other hand, as these codes become longer, their performance/complexity tradeoff improves and approaches that of $\mathbb{C}_{\mathcal{G}}$.

[1]There are five inequivalent $(32, 16, 8)$ codes. This $(32, 16, 8)$ code has been kindly identified as the code f_2^{16+} of [10] by Sloane [56], namely, the last code C_{85} in [52, Table IV]. In other words, it is neither the $(32, 12, 8)$ quadratic-residue code nor the $(32, 16, 8)$ Reed–Muller code.

IV. The Hexacode

The *hexacode* H_6 is a remarkable $(6, 3, 4)$ linear code over the quaternary field $\mathbb{F}_4 = \{0, 1, \omega, \overline{\omega}\}$. It is the basis for the Miracle Octad Generator (MOG) construction of the Golay code outlined in the previous section, and may be also used to construct the Leech lattice. Decoding the hexacode is a key step in the most efficient algorithms known [62], [63], [60] for decoding the Golay code and the Leech lattice. For a full exposition of the properties of H_6, see [11, Ch. 11].

In this section, we investigate tail-biting trellises for H_6. We will show that a group tail-biting trellis for the hexacode (over the additive group of $\mathbb{F}_4$) is much nicer than any linear (over $\mathbb{F}_4$) trellis for H_6. We note that although the hexacode and the Golay code are closely related, there appears to be no relation whatsoever between their minimal tail-biting trellises.

In the standard coordinate system for H_6, it is the image of the five hexacodewords $(00\ 00\ 00), (00\ 11\ 11)$, $(01\ 01\ \omega\overline{\omega})$, $(\omega\overline{\omega}\ \omega\overline{\omega}\ \omega\overline{\omega})$, and $(11\ \omega\omega\ \overline{\omega}\overline{\omega})$ under the following symmetries:

- scalar multiplication by any nonzero element;
- bodily permutation of the three pairs in any way;
- interchanging the two components in any two pairs.

Since the hexacode is maximum-distance separable (MDS), the complexity of a minimal conventional trellis is the "worst possible," regardless of the order of the time axis [18]. The corresponding state-complexity profile is $\{1, 4, 16, 64, 16, 4, 1\}$. In particular, there must be $\mathcal{S}_{\text{mid}} = 64$ states at the midpoint of any conventional trellis for H_6.

In view of the results of the previous section, one might be tempted to try to find a generator matrix for H_6 of the following form:

$$\begin{bmatrix} \star\star & \star\star & 00 \\ 00 & \star\star & \star\star \\ \star\star & 00 & \star\star \end{bmatrix}. \tag{8}$$

Such a generator matrix could be used to construct a tail-biting trellis with state-complexity profile $\{4, 16, 4, 16, 4, 16\}$. However, we now show that such a generator matrix for H_6 does not exist.

Proposition 5: The hexacode H_6 does not have a generator matrix of the form (8), under any coordinate permutation.

Proof: We show that there is no set of three linearly independent hexacodewords of weight 4, such that their zeroes jointly cover all six coordinates. The weight-4 codewords in H_6 are the nine images of $(00\ 11\ 11)$ and the 36 images of $(01\ 01\ \omega\overline{\omega})$ under the symmetries above. It suffices to consider only the three images of $(00\ 11\ 11)$ and the 12 images of $(01\ 01\ \omega\overline{\omega})$ under the latter two symmetries, since their scalar multiples will have zeros in the same positions. It is not difficult to see that, up to these symmetries, only the following sets of three hexacodewords have zeroes that jointly cover all

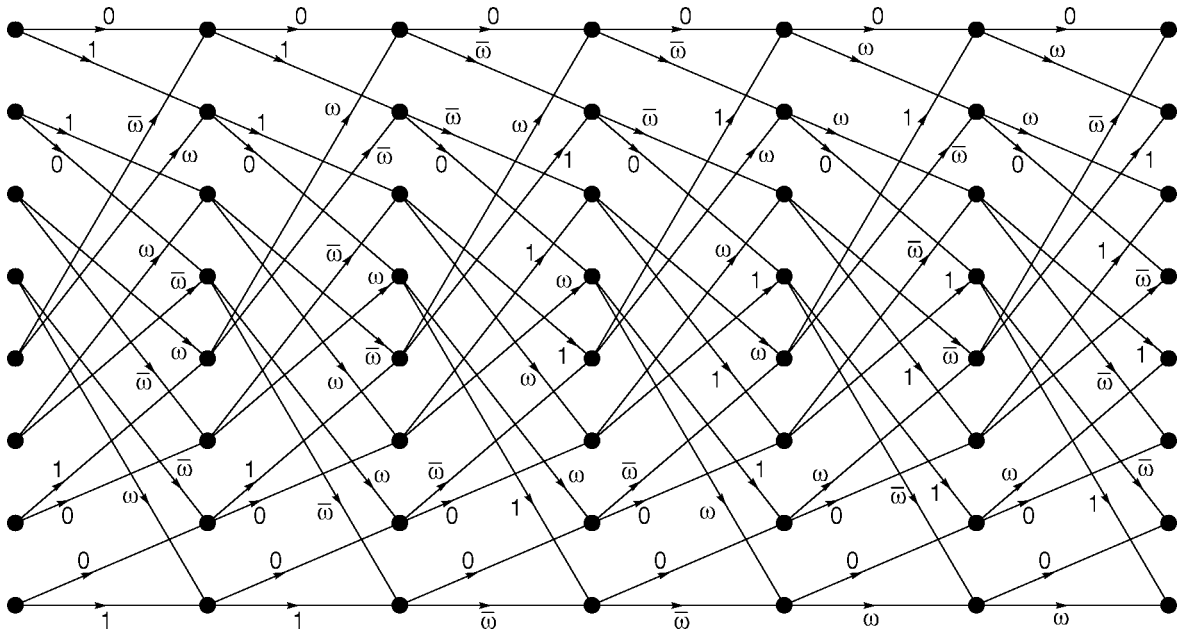


Fig. 7. 8-state group tail-biting trellis for the hexacode over V.

six coordinates:

$$\begin{Bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{Bmatrix} \quad \begin{Bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & \omega & \overline{\omega} \\ 1 & 0 & 1 & 0 & \omega & \overline{\omega} \end{Bmatrix}$$

$$\begin{Bmatrix} 0 & 1 & 0 & 1 & \omega & \overline{\omega} \\ 1 & 0 & \omega & \overline{\omega} & 1 & 0 \\ \overline{\omega} & \omega & 1 & 0 & 0 & 1 \end{Bmatrix}.$$

None of these sets is independent. In the first two cases, the three words sum to zero, while in the last case we have $(01\ 01\ \omega\overline{\omega}) + \omega(10\ \omega\overline{\omega}\ 10) + \overline{\omega}(\overline{\omega}\omega\ 10\ 01) = (00\ 00\ 00)$. □

However, just a slight modification of the structure in (8) suffices to produce a generator matrix for H_6 over $\mathbb{F}_4$. In particular, the hexacode does have the following generator matrix:

$$\begin{bmatrix} 11 & 11 & 00 \\ 00 & 11 & 11 \\ 10 & 01 & \overline{\omega}\omega \end{bmatrix}.$$

This generator matrix specifies a linear tail-biting trellis for H_6 over $\mathbb{F}_4$, with a less symmetrical state-complexity profile $\{4, 4, 4, 16, 16, 16\}$. In view of Proposition 5, this profile is the best possible. Note that it meets the cut-set lower bounds $|S_j|\,|S_{j+3}| \geq 64$ for all j. However, it fails to meet the square-root lower bound $\mathcal{S}_{\max} \geq 8$.

As noted in Example 1 of Section II, a linear block code $\mathbb{C}$ over the quaternary field $\mathbb{F}_4$ is also a group code over the additive group $V = \{0, 1, \omega, \overline{\omega}\}$. Moreover, a group tail-biting trellis for $\mathbb{C}$ may be simpler than any linear trellis for $\mathbb{C}$. As a group code, the hexacode is isomorphic to $\mathbb{Z}_2^6$. In particular, H_6 may be represented as the set of 64 linear combinations *with binary coefficients* of the following six hexacodewords:

$$\boldsymbol{G}_6 \stackrel{\text{def}}{=} \begin{Bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & \overline{\omega} & \omega & 1 & 0 \\ 0 & 0 & \overline{\omega} & \overline{\omega} & \overline{\omega} & \overline{\omega} \\ \overline{\omega} & 0 & 0 & \overline{\omega} & \omega & 1 \\ \omega & \omega & 0 & 0 & \omega & \omega \\ 1 & \overline{\omega} & \omega & 0 & 0 & \omega \end{Bmatrix}. \tag{9}$$

This generator set specifies an 8-state group tail-biting trellis for H_6 over V with state-complexity profile $\{8, 8, 8, 8, 8, 8\}$. This profile is minimal everywhere, by the square-root lower bound. The resulting trellis is illustrated in Fig. 7. As may be concluded from the foregoing discussion, it is both simpler and more regular than the best possible linear tail-biting trellis for H_6 over $\mathbb{F}_4$.

The generator set $\boldsymbol{G}_6$ is by no means unique. There are in fact 294 such generator sets, of which 34 are inequivalent under the automorphisms of H_6, as we have determined by a straightforward classification. However, the set $\boldsymbol{G}_6$ is in some sense the "nicest." It is "quasi-constacyclic," meaning that $\boldsymbol{G}_6$ is invariant under the cyclic shift by two positions, followed by a multiplication by ω. On the other hand, none of the 294 different generator sets exhibits any further symmetries of the hexacode, not even its linearity.

The trellis of Fig. 7 is again structurally invariant, but not cyclic. It is both 3-controllable and 3-observable. That is, any state may be reached from any other state in three time units, and any three consecutive symbols determine a unique trellis path. There is a unique hexacodeword corresponding to any state pair $(s_j, s_{j+3}) \in S_j \times S_{j+3}$, for any $j \in \mathbb{Z}_6$.

By multiplying the second 2-symbol block in (9) by ω and the third 2-symbol block by $\overline{\omega}$, we obtain another generator

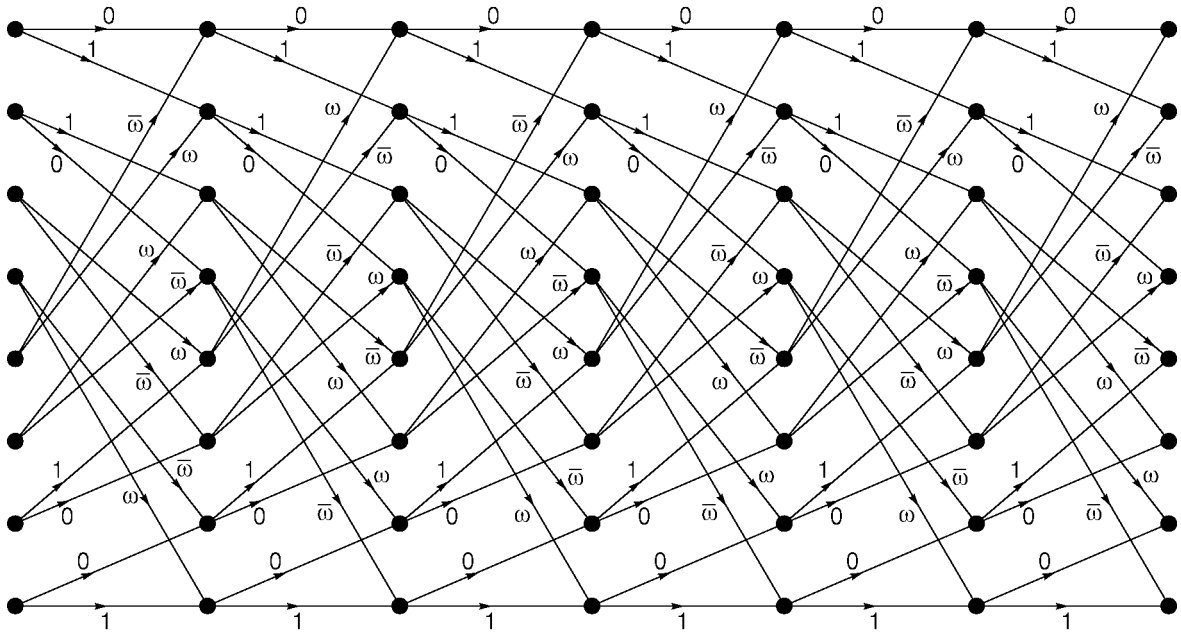


Fig. 8. 8-state period-2 group tail-biting trellis for an equivalent $(6,3,4)$ linear code.

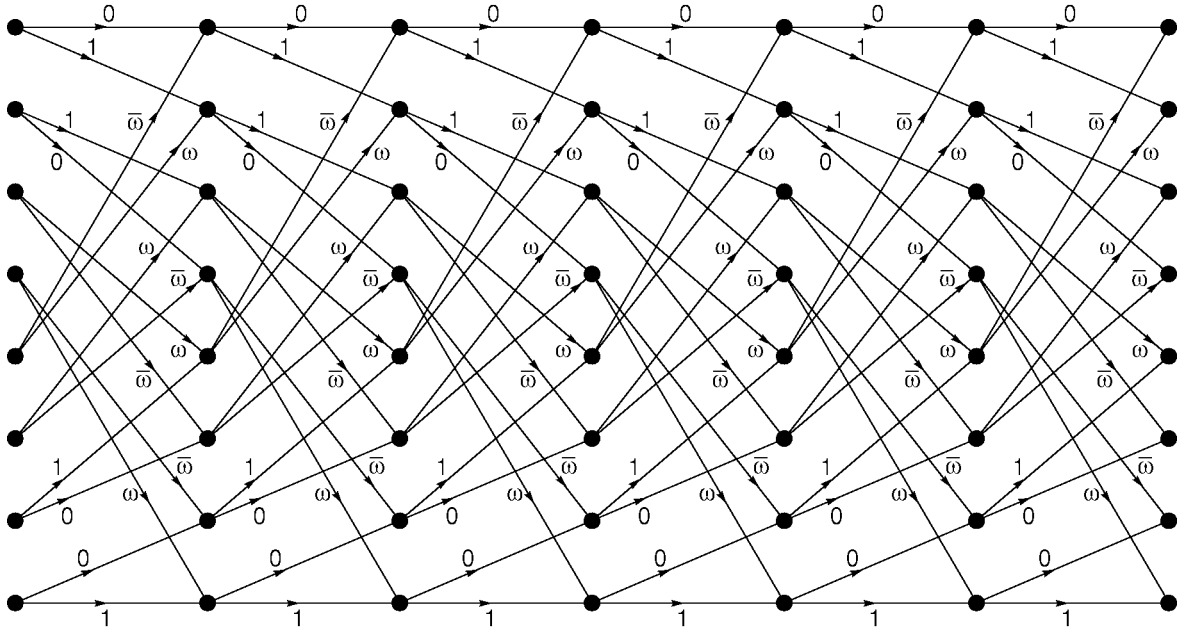


Fig. 9. 8-state cyclic tail-biting trellis for the group code E_6.

set, given by

$$G_6' \stackrel{\text{def}}{=} \begin{Bmatrix} 1 & 1 & \omega & \omega & 0 & 0 \\ 0 & 1 & 1 & \overline{\omega} & \overline{\omega} & 0 \\ 0 & 0 & 1 & 1 & \omega & \omega \\ \overline{\omega} & 0 & 0 & 1 & 1 & \overline{\omega} \\ \omega & \omega & 0 & 0 & 1 & 1 \\ 1 & \overline{\omega} & \overline{\omega} & 0 & 0 & 1 \end{Bmatrix}. \tag{10}$$

This set generates a monomially equivalent[2] $(6,3,4)$ linear code H_6' that is periodically time-varying with period 2. The corresponding period-2 trellis is illustrated in Fig. 8.

[2]Monomially equivalent means equivalent up to coordinate permutation and coordinate multiplication by scalars.

Finally, by conjugating the second components of each block in (10), we obtain a third generator set G_6'' consisting of the six cyclic shifts of the single generator $(11\ \omega\overline{\omega}\ 00)$. Thus G_6'' evidently generates a cyclic $(6,3,4)$ group code E_6 over V. However, E_6 is no longer a linear code over $\mathbb{F}_4$, since such conjugation does not preserve linearity. This hexacode-equivalent cyclic group code E_6 was previously found by Ran and Snyders [53]. Its trellis, shown in Fig. 9, is both minimal and cyclic. The dynamical structure is the same as in Figs. 7 or 8, but the labeling is even more symmetrical, and the properties of E_6 are easier to deduce from its trellis.

If H_6 is "unwrapped" in the same way as we "unwrapped" the Golay code in the previous section, then we obtain

a periodically time-varying "convolutional hexacode" $\mathbb{C}_H$ with period 6 and free-distance $d = 4$. If the group code E_6 is unwrapped, then we obtain a time-invariant convolutional group code $\mathbb{C}_E$ over V, generated by the shifts of $(\cdots 0011\omega\overline{\omega}\cdots)$. The free distance of $\mathbb{C}_E$ is also $d = 4$. Unwrapping H_6' yields a version $\mathbb{C}_{H'}$ of $\mathbb{C}_E$ in which alternate symbols are conjugated. In all cases, the input group (cf. [22]) is isomorphic to $\mathbb{Z}_2$ and the output group is $V \simeq \mathbb{Z}_2^2$. Thus the rate and the redundancy are both equal to 1 bit per symbol.

We can further map the symbols in $\mathbb{C}_H, \mathbb{C}_{H'}$ or $\mathbb{C}_E$ onto binary 2-tuples, or more generally onto the four subsets of any four-way partition. Some of the images of $\mathbb{C}_H, \mathbb{C}_{H'}, \mathbb{C}_E$ obtained in this way are quite good codes. In particular

- Consider the map $\{0, 1, \omega, \overline{\omega}\} \rightarrow \{00, 11, 01, 10\}$. If the code symbols of either $\mathbb{C}_{H'}$ or $\mathbb{C}_E$ are mapped onto $\mathbb{F}_2^2$ using this map, then we obtain an 8-state rate-$1/2$ binary linear time-invariant convolutional code with free distance $d = 6$. This distance is the best possible. The number of minimum-weight codewords is $N_6 = 3$, which is not quite the best possible [12]. The image of the "convolutional hexacode" $\mathbb{C}_H$ under this map has free distance of only $d = 4$.
- Let A_2 denote the two-dimensional hexagonal lattice. Then (cf. [17]) the Coxeter–Todd lattice K_{12} may be constructed by mapping the symbols of the hexacode H_6, or of the group code E_6, onto the four cosets of $2A_2$ in A_2, according to any isomorphism from V to the quotient group $A_2/2A_2$. The image of the convolutional hexacode $\mathbb{C}_H$, or of $\mathbb{C}_E$, under the same map is an 8-state "Coxeter–Todd trellis code," with the same nominal coding gain of $4/\sqrt{3}$, or 3.63 dB, but with a lower error coefficient. The error coefficient is 54 per two dimensions for the image of $\mathbb{C}_E$, rather than 126 per two dimensions for the Coxeter–Todd lattice K_{12}.

V. Minimal Tail-Biting Trellises for Other Short Codes

We now construct minimal tail-biting trellises for the $(8, 4, 4)$ extended binary Hamming code, the $(4, 2, 3)$ tetracode over $\mathbb{F}_3$, the $(4, 2, 3)$ quadracode over $\mathbb{F}_4$, and the $(8, 4, 4)$ octacode over $\mathbb{Z}_4$. The trellises we construct for the $(4, 2, 3)$ codes are minimal by the square-root lower bound. To prove the minimality of the trellises for the $(8, 4, 4)$ codes, we first develop another lower bound based on the total generator span.

A. The Total Span Bound

In this subsection, we give another lower bound on $\mathcal{S}_{\max}$ that is based on the concept of the total active length of a generator matrix.

The *active length* θ_i of a generator $\boldsymbol{g}_i$ is defined as one less than the cardinality of its (circular) span. Thus θ_i is the "length" of the (circular) active interval of $\boldsymbol{g}_i$. For a set of k generators, we define the *total active length* as the sum $\Theta = \theta_1 + \theta_2 + \cdots + \theta_k$. In the linear case, over a field $\mathbb{F}_q$, a generator of active length θ_i contributes one dimension to the state space, over a time interval of length θ_i. Hence the product of all m state-space sizes is given by

$$\prod_{j=0}^{m-1} |S_j| = q^{\theta_1} q^{\theta_2} \cdots q^{\theta_k} = q^{\Theta}.$$

The maximum state-space size $\mathcal{S}_{\max}$ is therefore at least $q^{\Theta/m}$. Furthermore, in a linear tail-biting trellis over $\mathbb{F}_q$, the state-space dimension must be an integer at all times, which implies that $\mathcal{S}_{\max} \geq q^{\lceil \Theta/m \rceil}$. Notice that this is not necessarily true for group trellises. In summary, we have proved the following.

Proposition 6 (Total Span Bound): The product of the state-space sizes in an m-section linear tail-biting trellis for a linear code $\mathbb{C}$ over $\mathbb{F}_q$, based on a generator matrix with total active length Θ, is lower-bounded by q^{Θ}. Consequently, $\mathcal{S}_{\max}$ is lower-bounded by $q^{\lceil \Theta/m \rceil}$ for linear trellises, and by $q^{\Theta/m}$ for group trellises.

By definition, the span of a generator includes all of its nonzero positions. Hence, for a code with minimum distance d we must have $\theta_i \geq d - 1$ for all i.

Corollary 7: If $\mathbb{C}$ is an (n, k, d) linear code over $\mathbb{F}_q$, then any n-section linear tail-biting trellis for $\mathbb{C}$ satisfies

$$\prod_{j=0}^{n-1} |S_j| \geq q^{k(d-1)} \tag{11}$$

$$\mathcal{S}_{\max} \geq q^{\lceil R(d-1) \rceil} \tag{12}$$

where $R = k/n$ is the rate of $\mathbb{C}$. For group trellises, we have $\mathcal{S}_{\max} \geq q^{R(d-1)}$.

Proof: The active length of each generator must be at least $d - 1$, so the total active length Θ must be at least $k(d-1)$. □

We note that the total span lower bound is well known [29], [39] in the context of conventional trellises. However, Proposition 6 and Corollary 7 show that this bound holds essentially without change for tail-biting trellises. In particular, it follows immediately from Corollary 7, in conjunction with the argument of [38], that the maximum state-space size $\mathcal{S}_{\max}$ grows exponentially fast in any sequence of tail-biting trellises for asymptotically good codes.

We can use Corollary 7 to obtain tight lower bounds for MDS codes. Since for an MDS code we have $d = n - k + 1$, the bound of (12) becomes

$$\mathcal{S}_{\max} \geq q^{\lceil nR(1-R) \rceil}.$$

If $R = 1/2$, then the state complexity of a conventional trellis for an MDS code at the midpoint is $q^{n/2}$ regardless of the order of the time axis, and Corollary 7 reduces to the square-root bound. The quaternary $(2, 1, 2)$ and $(6, 3, 4)$ codes are two examples where $nR(1 - R)$ is not an integer, so that the lower bounds for the linear case and the group case are genuinely different. Moreover, we have seen that both bounds are achievable for these two codes.

Corollary 7 also shows that the minimal conventional trellis for an $(n, 1, n)$ repetition code or an $(n, n-1, 2)$ single-parity-check code cannot be improved upon if state spaces are vector

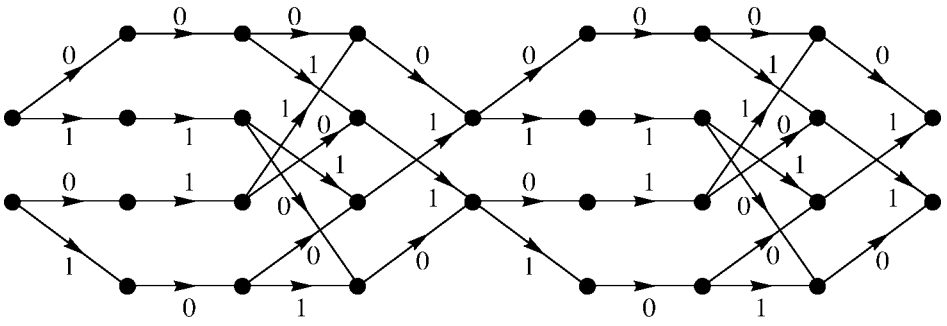


Fig. 10. 8-section minimal tail-biting trellis for the $(8,4,4)$ Hamming code.

spaces over $\mathbb{F}_q$. The minimum total active length is $\Theta = n-1$ in either case and, therefore, the best possible state-complexity profile is $\{1, q, q, \cdots, q\}$. However, the quaternary $(2,1,2)$ code of Example 1 shows that improvements are possible even for such apparently irreducible codes, by the use of group trellises.

B. The $(8,4,4)$ Hamming Code

The $(8,4,4)$ extended binary Hamming (or first-order Reed–Muller, or biorthogonal) code $\mathbb{C}_8$ is a notable self-dual code. It is used in the Turyn construction of the Golay code (see Appendix B). The set of all integer 8-tuples that are congruent modulo 2 to a codeword of $\mathbb{C}_8$ is the well-known Gosset lattice E_8 (cf. [11]).

The minimal conventional trellis for this code has state-complexity profile $\{1,2,4,8,4,8,4,2,1\}$, and therefore the square-root lower bound is $\mathcal{S}_{\max} \geq 2$. However, using the more general cut-set lower bound, with cuts into index sets of size 3 and 5, we conclude that a tail-biting trellis for $\mathbb{C}_8$ with constant state-complexity profile $\{2,2,2,2,2,2,2,2\}$ cannot exist. On the other hand, this argument does not rule out the existence of a trellis with state-complexity profile $\{2,4,2,4,2,4,2,4\}$. Using the total span bound of Proposition 6, we will show that in fact the best possible tail-biting state-complexity profile for the Hamming code is $\{2,4,4,4,2,4,4,4\}$, which meets the square-root bound in only one position.

Lemma 8: The minimum total active length of a generator matrix for $\mathbb{C}_8$ is $\Theta = 14$.

Proof: Since the minimum weight in $\mathbb{C}_8$ is 4, the active length of all generators must be at least 3. Suppose three generators have active length 3. Then all must have weight 4 and up to cyclic permutations must be $(11\ 11\ 00\ 00), (00\ 11\ 11\ 00)$, and $(00\ 00\ 11\ 11)$. These three vectors generate an $(8,3,4)$ subcode of $\mathbb{C}_8$. The $(8,4,4)$ Hamming code is the union of this subcode with its coset $(8,3,4)+(01\ 01\ 01\ 01)$. It is easy to see that all words in this coset have (circular) active length at least 5; e.g., $(01\ 01\ 10\ 10)$ or $(10\ 10\ 01\ 01)$. Alternatively, if only two of the generators have active length 3, then the other two must have active length at least 4. □

Theorem 9: The minimum product of state-space sizes in an 8-section trellis for the $(8,4,4)$ Hamming code is 2^{14}. Furthermore, $\mathcal{S}_{\max} \geq 4$.

Proof: This follows immediately from Proposition 6. Since the binary field $\mathbb{F}_2$ has no nontrivial multiplicative structure, there is no distinction between the linear case and the group case. □

The product of state-space sizes in the minimal conventional trellis is indeed 2^{14}, but $\mathcal{S}_{\max} = 8$. However, the following period-4 generator matrix

$$\boldsymbol{G}_8 \stackrel{\text{def}}{=} \begin{bmatrix} 11 & 11 & 00 & 00 \\ 00 & 11 & 01 & 10 \\ 00 & 00 & 11 & 11 \\ 01 & 10 & 00 & 11 \end{bmatrix} \tag{13}$$

leads to a tail-biting trellis with the optimal state-complexity profile $\{2,4,4,4,2,4,4,4\}$. This trellis is illustrated in Fig. 10. A nonisomorphic tail-biting trellis for $\mathbb{C}_8$ with the same state-complexity profile was found in [30]; it is shown in [31] that exactly two such trellises exist.

There is a 4-section version of this trellis with state-complexity profile $\{2,4,2,4\}$ and period 2. This version is shown in Fig. 11. Of course, there is also a 2-section cyclic tail-biting trellis with state-complexity profile $\{2,2\}$, which meets the square-root lower bound.

By unwrapping the generators for $\mathbb{C}_8$ in (13), in a manner similar to the unwrapping of the Golay and hexacode generators in the previous two sections, we obtain generators for a "Hamming convolutional code" $\mathbb{C}_{\mathcal{H}}$. This code is generated by

$$\begin{Bmatrix} 11 & 11 & 00 & 00 \\ 00 & 11 & 01 & 10 \end{Bmatrix}.$$

Thus $\mathbb{C}_{\mathcal{H}}$ may be characterized as a period-2 rate-$1/2$ code with alternately two and four states, whose trellis is the indefinite repetition of Fig. 11.

Alternatively, $\mathbb{C}_{\mathcal{H}}$ may be regarded as a time-invariant 2-state rate-$2/4$ partial-unit-memory code, with generators $(\mathbf{1}|\mathbf{0})$ and $(\boldsymbol{g}_1|\boldsymbol{g}_2)$, where $\mathbf{0}$ and $\mathbf{1}$ are the all-zero and all-one 4-tuples, and $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are different generators for the four cosets of the $(4,1,4)$ code in the $(4,3,2)$ code. This generator set resembles that of the Lee–Lauer code. Indeed, $\mathbb{C}_{\mathcal{H}}$ was discovered by Lauer [40], and is the best known 2-state rate-$2/4$ code. Notice that the best (cf. [12]) time-invariant 2-state rate-$1/2$ code has free distance only $d = 3$.

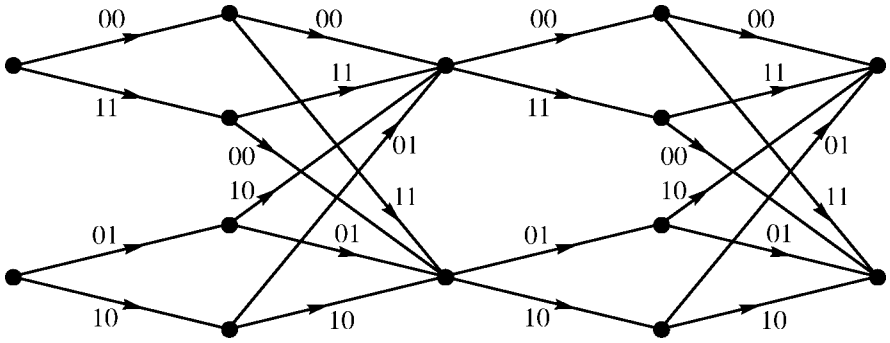
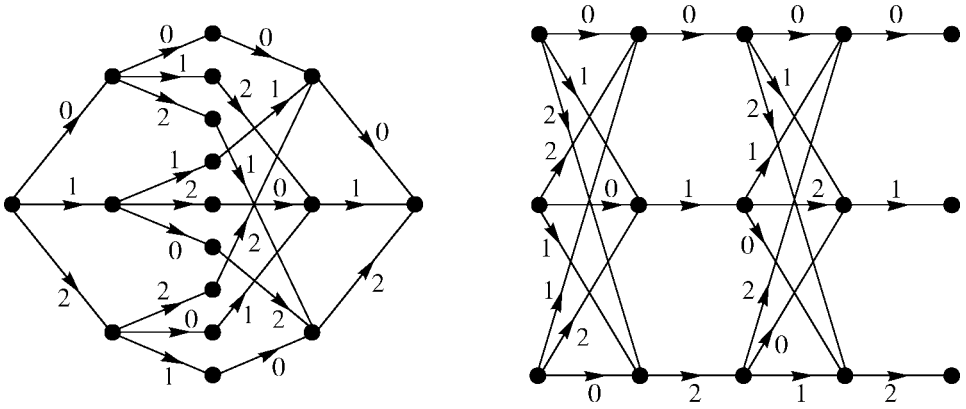


Fig. 11. $2/4$-state period-2 tail-biting trellis for the $(8,4,4)$ Hamming code.



(a) (b)

Fig. 12. 9-state conventional and 3-state tail-biting trellises for the $(4,2,3)$ tetracode.

Using straightforward flow-graph techniques, the distance-generating function of $\mathbb{C}_{\mathcal{H}}$ is easily determined to be

$$A(z) = z^4 + \frac{4z^4}{1-2z^2} = 5z^4 + 8z^6 + 16z^8 + \cdots.$$

Thus $d = 4$, and the number of minimum-weight sequences is $N_4 = 5$. Clearly $\mathbb{C}_{\mathcal{H}}$ is even, since all generators have weight 4, but not doubly-even since it is not self-dual. (This shows that a convolutional code derived from a tail-biting trellis for a Type II block code need not be Type II.)

A version of the Gosset lattice E_8 is obtained by mapping the bits in $\mathbb{C}_8$ to the two cosets of the even integer lattice $2\mathbb{Z}$ in the integer lattice $\mathbb{Z}$ (this is [11, Construction A]). Thus any trellis for $\mathbb{C}_8$ is also a trellis for E_8, including the tail-biting trellises of Figs. 10 and 11.

The image of $\mathbb{C}_{\mathcal{H}}$ under the same map is a 2-state four-dimensional trellis code analogous to E_8, with the same nominal coding gain of 2, or 3.01 dB, but with a slightly lower error coefficient. The error coefficient per two dimensions is only 44 for the image of $\mathbb{C}_{\mathcal{H}}$, rather than 60 for E_8. This trellis code was mentioned in [16].

C. The $(4,2,3)$ Tetracode

The $(4,2,3)$ tetracode T_4 is a self-dual linear MDS code over the ternary field $\mathbb{F}_3 = \{0,1,2\}$. It is used in an alternative construction of the Gosset lattice E_8 [11].

Since T_4 is MDS, the minimal conventional trellis has state-complexity profile $\{1,3,9,3,1\}$ regardless of the order of the time axis, and the square-root lower bound is $\mathcal{S}_{\max} \geq 3$. This bound is met with equality by the 3-state tail-biting trellis specified by the generators (1110) and (2011). Indeed, this tail-biting trellis has constant state-complexity profile $\{3,3,3,3\}$. The conventional and the tail-biting trellises for T_3 are illustrated in Fig. 12.

The tetracode cannot be made cyclic, or even quasi-cyclic. If the tail-biting trellis is unwrapped, we get a 3-state rate-$1/2$ time-varying linear convolutional code over $\mathbb{F}_3$ with free distance $d = 3$. This code is inferior to the best possible code, which has free distance $d = 4$.

Another version of the Gosset lattice E_8 is obtained by mapping the symbols in T_4 componentwise to the ternary quotient group $A_2/\theta A_2$, which represents a three-way partition of the hexagonal lattice A_2. In this way, any trellis for T_4 maps to a trellis for E_8, including the tail-biting trellis of Fig. 12. Thus the tetracode yields a 3-state tail-biting trellis for E_8, whereas $\mathcal{S}_{\max} \geq 4$ for any trellis based on $\mathbb{C}_8$, tail-biting or not. However, the trellis for E_8 obtained from the Hamming code may nonetheless be easier to decode.

D. The $(4,2,3)$ Quadracode

The $(4,2,3)$ quadracode Q_4 is an MDS linear code over the quaternary field $\mathbb{F}_4$. Since Q_4 is MDS, the minimal

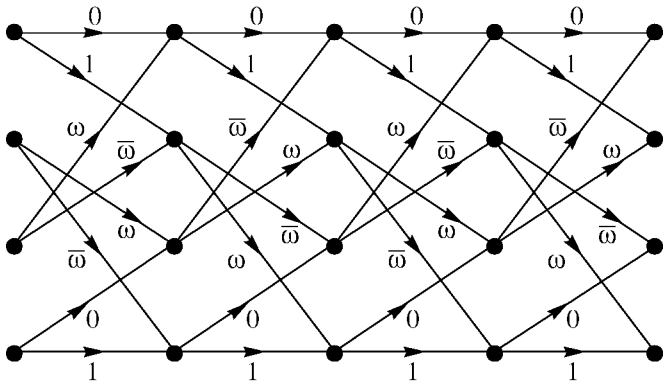


Fig. 13. 4-state group tail-biting trellis for the $(4, 2, 3)$ quadracode.

conventional trellis has state complexity profile $\{1, 4, 16, 4, 1\}$, and the square-root lower bound is $\mathcal{S}_{\max} \geq 4$. This lower bound can be met by a linear trellis for Q_4, using for example the generators $\{01\omega\overline{\omega}, \omega\overline{\omega}01\}$. This yields a 4-state tail-biting trellis similar to that of Fig. 12. However, a more symmetrical trellis is obtained by regarding Q_4 as a group code. For example, the following generator set:

$$\begin{Bmatrix} 0 & 1 & \omega & \overline{\omega} \\ \omega & 0 & 1 & \overline{\omega} \\ \omega & \overline{\omega} & 0 & 1 \\ 1 & \overline{\omega} & \omega & 0 \end{Bmatrix} \tag{14}$$

produces a period-2 group tail-biting trellis for Q_4, which has constant state-complexity profile $\{4, 4, 4, 4\}$. This minimal trellis is illustrated in Fig. 13.

The quadracode can be made cyclic by conjugating every other symbol to obtain the $(4, 2, 3)$ group code E_4, generated by the four cyclic shifts of $(01\ \omega\omega)$. However, E_4 is no longer linear over $\mathbb{F}_4$. The cyclic group code E_4 was also previously found by Ran and Snyders [53].

The convolutional code obtained by unwrapping either Q_4 or E_4 is a quaternary 4-state code with 1 bit of redundancy per symbol and free distance $d = 3$. If this code is mapped onto $\mathbb{F}_2^2$ via the map $\{0, 1, \omega, \overline{\omega}\} \rightarrow \{00, 11, 01, 10\}$, then we obtain a 4-state rate-$1/2$ binary linear convolutional code with free distance $d = 4$. This is inferior to the best possible [12] distance $d = 5$.

E. The $(8, 4, 4)$ Octacode

The $(8, 4, 4)$ octacode $\mathcal{O}_8$ is a self-dual linear code over the quaternary ring $\mathbb{Z}_4$. It has minimum Lee distance 6, and its binary image under the Gray map $\{0, 1, 2, 3\} \rightarrow \{00, 01, 11, 10\}$ is the Nordstrom–Robinson code [21], [24]. It is used in [11, Ch. 24] as the glue code in the "holy construction" of the Leech lattice from eight copies of the face-centered cubic lattice. The projection of the octacode onto $\mathbb{Z}_2$ is the $(8, 4, 4)$ binary Hamming code.

The minimal conventional trellis for $\mathcal{O}_8$, under all possible permutations of the time axis, was constructed in [17] and [18]. This trellis has state complexity profile $\{1, 4, 16, 64, 64, 64, 16, 4, 1\}$, so the square-root lower bound is $\mathcal{S}_{\max} \geq 8$. However, we now show that this is not tight.

The Total Span Lower Bound for the Octacode: Given a set $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_4\}$ of four generators for the octacode over $\mathbb{Z}_4$, it can be expressed as the set of all *binary* linear combinations of the eight generators $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_4, 2\boldsymbol{g}_1, 2\boldsymbol{g}_2, 2\boldsymbol{g}_3, 2\boldsymbol{g}_4\}$, which determine a corresponding trellis. Note that the span of $2\boldsymbol{g}_i$ is not necessarily equal to the span of $\boldsymbol{g}_i$; it may be[3] a proper subset.

The four generators $\{2\boldsymbol{g}_1, 2\boldsymbol{g}_2, 2\boldsymbol{g}_3, 2\boldsymbol{g}_4\}$ generate the code $2\mathcal{O}_8$, a subcode of $\mathcal{O}_8$ isomorphic to the $(8, 4, 4)$ binary Hamming code. Therefore, by Lemma 8, the minimum total active length of these generators is 14. The generators $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_4\}$ are odd and therefore have minimum Hamming weight 5 and active length at least 4. Hence, their total active length is at least 16. The span bound of Proposition 6 thus shows that the product of the state-space sizes in any tail-biting trellis for $\mathcal{O}_8$ is at least 2^{30}. Notice that this bound is met by the minimal conventional trellis.

A minimal tail-biting trellis for $\mathcal{O}_8$ with state-complexity profile $\{8, 16, 16, 16, 8, 16, 16, 16\}$ may be constructed as the product of the two-state trellises corresponding to the binary linear combinations of the following set of eight generators:

$$\begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \boldsymbol{g}_3 \\ \boldsymbol{g}_4 \\ 2\boldsymbol{g}_1 \\ 2\boldsymbol{g}_2 \\ 2\boldsymbol{g}_3 \\ 2\boldsymbol{g}_4 \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} 13 & 11 & 00 & 02 \\ 00 & 31 & 21 & 10 \\ 00 & 02 & 13 & 11 \\ 21 & 10 & 00 & 13 \\ 22 & 22 & 00 & 00 \\ 00 & 22 & 02 & 20 \\ 00 & 00 & 22 & 22 \\ 02 & 20 & 00 & 22 \end{bmatrix}. \tag{15}$$

The trellis with state-complexity profile $\{8, 16, 16, 16, 8, 16, 16, 16\}$ is indeed minimal, by the total span lower bound. Note that the set $\{2\boldsymbol{g}_1, 2\boldsymbol{g}_2, 2\boldsymbol{g}_3, 2\boldsymbol{g}_4\}$ is obtained by "lifting" our earlier generator matrix $\boldsymbol{G}_8$ for the $(8, 4, 4)$ Hamming code, given in (13).

The unlabeled structure of the corresponding minimal tail-biting trellis is shown in Fig. 14. The structure has period 4, but the self-duality of $\mathcal{O}_8$ prevents a period-4 labeling.

The convolutional code obtained by unwrapping $\mathcal{O}_8$ is a self-dual $\mathbb{Z}_4$-linear convolutional code with 1 bit of redundancy per symbol, minimum Hamming distance $d = 4$, and minimum Lee distance $d = 6$. If this code is mapped onto $\mathbb{F}_2^2$ via the Gray map, then we obtain a nonlinear 8/16-state rate-$1/2$ binary convolutional code with period 4 and free distance $d = 6$. This code is inferior to the Golay and Lee–Lauer convolutional codes discussed in Section III-B.

The Nordstrom–Robinson code NR_{16} is the image of $\mathcal{O}_8$ under the Gray map, and therefore any trellis for $\mathcal{O}_8$ is also a trellis for NR_{16}. The 8-section tail-biting trellis of Fig. 14 may be minimal, although a tail-biting trellis for NR_{16} with constant state-complexity profile $\{8, 8, 8, 8, 8, 8, 8, 8\}$ would not contradict any known bounds.

[3] As Trott [58] has pointed out, the span of $2\boldsymbol{g}_i$ need not even be chosen as a subset of the span of $\boldsymbol{g}_i$, in which case the resulting trellis is not necessarily a group trellis. Consider, for example, the code over $\mathbb{Z}_4$ generated by all binary linear combinations of the generator $\boldsymbol{g} = (2112)$ with span $[3, 2]$ and the generator $2\boldsymbol{g} = (0220)$ with span $[2, 3]$, which yields a nongroup minimal trellis with $\mathcal{S}_{\max} = 2$. However, this kind of situation does not arise with $\mathcal{O}_8$.

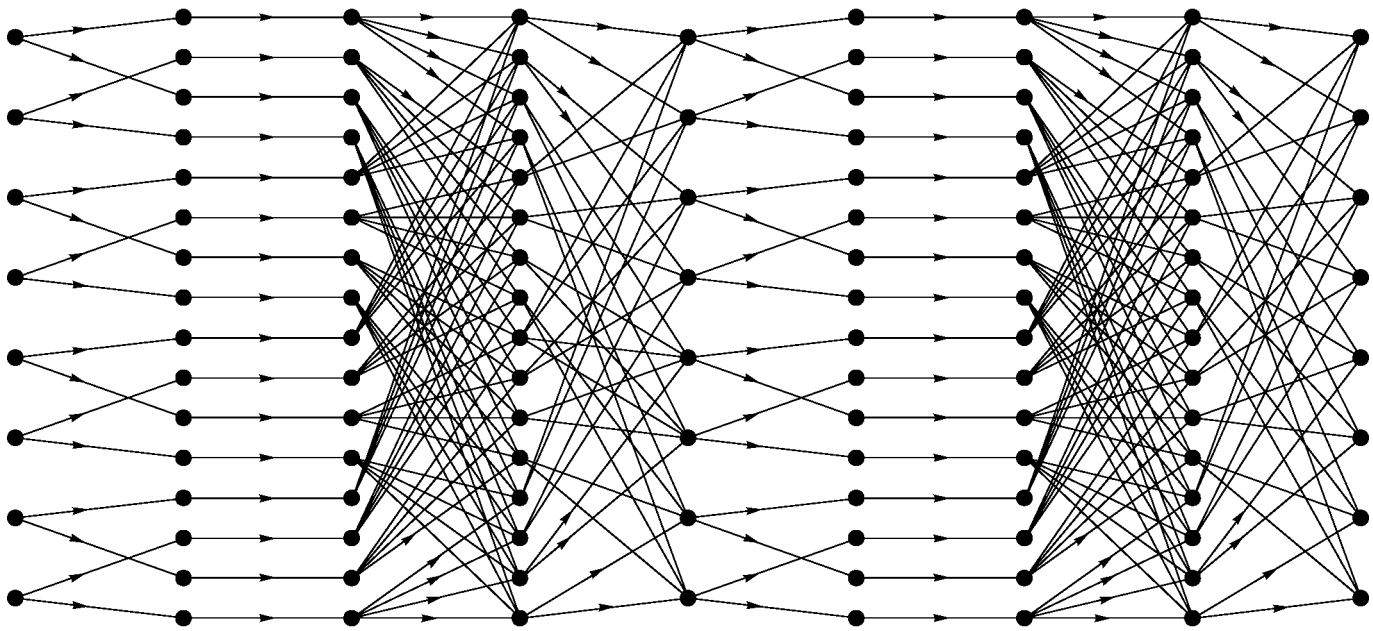


Fig. 14. Structure of the minimal $8/16$-state tail-biting trellis for the $(8,4,4)$ octacode.

VI. Decoding Algorithms for Tail-Biting Trellises

The only exact maximum-likelihood (ML) method known to us for decoding a tail-biting trellis is as follows [57]. The code represented by the trellis may be regarded as a union of subcodes, each subcode corresponding to the paths going through a given state in the state space S_0. Each of these subcodes may be decoded independently by the Viterbi algorithm. This produces $|S_0|$ candidate codewords; then the best of all these codewords is selected as the decoder output. The complexity of such a parallel decoding method is of the order of $|S_0|\mathcal{S}_{\max} \geq |S_0||S_{m/2}|$, which in view of the square-root lower bound cannot be less than the state complexity $\mathcal{S}_{\rm mid}$ at the midpoint of a conventional trellis. Indeed, we can also partition the code into subcodes according to any state space S_j, not necessarily S_0. However, a similar argument shows that the complexity of the resulting decoding method is again about the same as the complexity of decoding a conventional trellis for the same code.

Similarly, an exact symbol *a posteriori* probability (APP) decoding method is to compute the sum of products of likelihoods of codewords containing a given symbol value, for each of the subcodes of the previous paragraph independently, using the standard forward–backward (BCJR) algorithm for conventional trellises. Then one has to sum the resulting sums over all subcodes, to compute the total sum of such products [27, Sec. 7.3].

The following approximate method has been used by many previous authors [8], [57], [37], [44], [70], [65]. Initialize all metrics at the states in S_0 to zero. Using the branch metrics corresponding to the received symbols, decode with the Viterbi algorithm, starting at S_0, and going around and around the tail-biting trellis. Stop the algorithm after a preset number of cycles, or whenever a definite "winner" path has emerged. Experiments have shown that, for many codes at moderate-to-high signal-to-noise ratios, fewer than two decoding cycles often suffice.

The work of Wiberg [67] and others [68], [20], [49], [35], [23], [2], [36] has focussed attention on a class of iterative decoding algorithms generically known as the *sum-product algorithm*. On a graph with cycles, such as a tail-biting trellis, these decoding algorithms are only approximate, but may be much more efficient than exact algorithms. For example, the "round-and-round Viterbi decoding" method discussed above is an instance of the *min-sum form* of this algorithm, when applied to a tail-biting trellis. Weiss [66] and others have shown that if this algorithm converges, then it must converge to the maximum-likelihood codeword. Failure to converge is generally due to multicycle *pseudocodewords* that satisfy all local trellis constraints and have a better average log-likelihood weight per cycle than any valid codeword.

Similarly, the real sum-product form of this algorithm is a "round-and-round bidirectional BCJR" algorithm that computes approximate symbol APP's on tail-biting trellises. Anderson and Hladik [4] and others have shown, using the Perron–Frobenius theorem, that this algorithm always converges in a strong sense. However, it is hard to say what it converges to; it can be strongly attracted to pseudocodewords.

Consequently, for iterative decoding on a tail-biting trellis it is important to know the properties of the pseudocodewords in the trellis. Wiberg [67, pp. 52–54] gives a formula for the effective weight of pseudocodewords on the additive white Gaussian noise channel. For the Golay tail-biting trellis, we conjecture that there exist no pseudocodewords with effective weight less than 8. This conjecture is supported by the simulation results of Reznik [54] and others, which show that iterative decoding achieves near-ML performance, and by the fact that no such pseudocodewords have been found in extensive searches. However, it would be nice to have a proof.

In contrast, on the binary-symmetric channel, the Golay tail-biting trellis has pseudocodewords of effective weight 6, as shown by Xu and McEliece [69]. In other words, three channel errors can cause decoding ambiguity. Thus iterative decoding is not near-ML on binary-symmetric channels.

In summary, iterative decoding of tail-biting trellises can be much more efficient than conventional ML or APP decoding. While not exact, it can often achieve near-optimum performance, although not always. Our understanding of the performance of iterative decoding for this simplest case of a graph with cycles has progressed considerably, but is still incomplete.

VII. Conclusions

This paper has shown that more "art" is involved in constructing minimal tail-biting trellises than minimal conventional trellises. In addition to the usual problem of finding the best coordinate permutation and the corresponding generators, we are free to choose generator spans. Most surprisingly, nonlinear trellises can be superior to linear trellises, even for linear codes.

Absent a theory of minimal realizations on a circular time axis $\mathbb{Z}_m$, we have nonetheless been able to construct provably minimal tail-biting trellises for some important (short) block codes. Recent work of Kötter and Vardy [30]–[32] supports the optimality of our generator-based product-construction methods for linear tail-biting trellises.

Our results show that representations of codes on graphs with cycles can be significantly simpler, from a complexity viewpoint, than conventional trellis representations. Iterative decoding methods on such graphs can thus yield near-optimal performance with reduced complexity. However, the behavior of iterative decoding is still not very well understood, even for the simple tail-biting case. While the importance of pseudocodewords has now been recognized, we know little about their properties, even for the Golay tail-biting trellis.

The interesting properties of the Golay convolutional code suggest that unwrapping the generators of efficient tail-biting representations of known good algebraic block codes may be a general method of finding good convolutional codes. This approach could yield a route to the long-sought goal of applying the powerful tools developed for constructing algebraic block codes to the construction of good convolutional codes. The most interesting such construction since the Golay convolutional code has been the discovery of a 256-state, rate-$1/2$, tail-biting trellis for the $(48, 24, 12)$ quadratic-residue code and of the corresponding rate-$1/2$, 256-state, Type II convolutional code by Kötter and Vardy [33]. Kötter and Vardy [30] also found a 16-state, rate-$1/2$, tail-biting trellis for $\mathbb{C}_{24}$, which is not isomorphic to the trellis exhibited here. Their trellis also has a period of 8 bits, but is not structurally invariant. Finally, Johannesson, Stahl, and Wittenmark [26] have found two other less regular rate-$4/8$, 16-state, tail-biting trellises for the Golay code which yield rate-$4/8$, 16-state, Type II convolutional codes with $d = 8$.

We believe that better understanding of tail-biting trellises, which constitute the simplest case of codes defined on graphs with cycles, may contribute to the understanding of codes defined on more general graphs. We hope that this paper will stimulate further work in this area.

Appendix A
The Leech Lattice

A well-known construction for the Leech lattice Λ_{24} is based on the Golay code and the binary $(24, 23, 2)$ single-parity-check and $(24, 1, 24)$ repetition codes. For more details, see [3], [11], and [17].

For the single-parity-check and repetition codes, it is easy to see (cf. Section V-A) that there is no simpler tail-biting trellis than the minimal conventional trellis, which has state-complexity profile $\{1, 2, 2, \cdots, 2\}$. However, it is possible to place the state space consisting of a single state anywhere we like. Taking the product of such two-state trellises for the $(24, 23, 2)$ and $(24, 1, 24)$ codes with our 16-state, 12-section, tail-biting trellis for the Golay code $\mathbb{C}_{24}$, we obtain a 12-section tail-biting trellis for Λ_{24} with state-complexity profile

$$\{32, 64, 64, \cdots, 64, 64, 32, 64, 64, \cdots, 64, 64\}. \tag{16}$$

Notice that the two positions with 32 states can be chosen arbitrarily in this trellis—these correspond to the positions of single states in the trellises for the $(24, 1, 24)$ and $(24, 23, 2)$ codes. Alternatively, we could obtain the profile $\{16, 64, \cdots, 64\}$.

It is known [19] that the minimum state complexity at the midpoint of a conventional trellis for the Leech lattice is $\mathcal{S}_{\text{mid}} = 729$, for a complex (ternary) construction, and $\mathcal{S}_{\text{mid}} = 1024$, for a real (binary) construction. The state-complexity profile of the tail-biting trellis in (16) meets the square-root lower bound based on the latter value of $\mathcal{S}_{\text{mid}}$ in only one position.

Nonetheless, this could be the optimal tail-biting trellis for Λ_{24}. On the other hand, our bounds do not rule out the existence of a 12-section tail-biting trellis for Λ_{24} with 32 states everywhere. Finding such a trellis, or proving that no such trellis exists, is an interesting open problem.

Appendix B
Lee–Lauer Codes

In this appendix we give a brief general description of codes of the Lee–Lauer type. We are grateful to a reviewer for a reference to a similar development by Abdel-Ghaffar, McEliece, and Solomon [1] in a 1991 JPL report. Here, we wish particularly to show the relation of the Lee–Lauer codes to the Turyn[4] or "cubing" construction [17] of the Golay code.

The Turyn construction [17], [45] of the Golay code is based on two versions of the $(8, 4, 4)$ binary extended Hamming code, say $(8, 4, 4)$ and $(8, 4, 4)^*$. These two equivalent Hamming codes have the property that their intersection is the $(8, 1, 8)$ repetition code and their direct sum is the $(8, 7, 2)$

[4] According to McEliece [47], a construction similar to the Turyn construction was known to Solomon as early as 1960. However, the standard name [45, p. 588] derives from a 1967 report by Assmus, Mattson, and Turyn [5].

single-parity-check code, namely,

$$(8,4,4) \cap (8,4,4)^* = (8,1,8) \tag{17}$$

$$(8,4,4) \oplus (8,4,4)^* = (8,7,2). \tag{18}$$

The Golay code is then the set of all vectors of the form $(\boldsymbol{a}+\boldsymbol{x}|\boldsymbol{b}+\boldsymbol{x}|\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{x})$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are codewords of the $(8,4,4)$ Hamming code, $\boldsymbol{x}$ is a codeword of the $(8,4,4)^*$ Hamming code, and $(\cdot|\cdot)$ denotes concatenation. Notice that this construction of $\mathbb{C}_{24}$ leads to a componentwise-optimal minimal conventional trellis for the Golay code [17].

In view of (17) and (18), there exist two sets of generators

$$\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3\} \subset \mathbb{F}_2^8$$

and

$$\{\boldsymbol{g}_1^*, \boldsymbol{g}_2^*, \boldsymbol{g}_3^*\} \subset \mathbb{F}_2^8$$

such that $\{\mathbf{1}, \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3\}$ generate the $(8,4,4)$ Hamming code, $\{\mathbf{1}, \boldsymbol{g}_1^*, \boldsymbol{g}_2^*, \boldsymbol{g}_3^*\}$ generate the equivalent $(8,4,4)^*$ Hamming code, while $\{\mathbf{1}, \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_1^*, \boldsymbol{g}_2^*, \boldsymbol{g}_3^*\}$ generate the $(8,7,2)$ code, where $\mathbf{1}$ denotes the all-one 8-tuple. For example, the following sets of generators have these properties:

$$\begin{aligned}\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3\} &= \{01010101, 00110011, 00001111\} \\ \{\boldsymbol{g}_1^*, \boldsymbol{g}_2^*, \boldsymbol{g}_3^*\} &= \{11011000, 10101100, 11000110\}.\end{aligned} \tag{19}$$

A Lee–Lauer-type 8-state rate-$4/8$ binary linear partial-unit-memory convolutional code[5] may then be generated by the four generators $(\mathbf{1}|\mathbf{0}), (\boldsymbol{g}_1|\boldsymbol{g}_1^*), (\boldsymbol{g}_2|\boldsymbol{g}_2^*)$, and $(\boldsymbol{g}_3|\boldsymbol{g}_3^*)$, where $\mathbf{0}$ denotes the all-zero 8-tuple. Specifically, for the generators given in (19), we obtain the following set of "trellis-oriented" generators:

$$\begin{bmatrix} \mathbf{1}|\mathbf{0} \\ \boldsymbol{g}_1|\boldsymbol{g}_1^* \\ \boldsymbol{g}_2|\boldsymbol{g}_2^* \\ \boldsymbol{g}_3|\boldsymbol{g}_3^* \end{bmatrix} = \begin{bmatrix} 11111111 & 00000000 \\ 01010101 & 11011000 \\ 00110011 & 10101100 \\ 00001111 & 11000110 \end{bmatrix}.$$

The resulting state-complexity profile, for 1-bit sections, is

$$\{\cdots, 8, 16, 32, 64, 64, 64, 32, 16, 8, \cdots\}.$$

This is easily shown to be the best possible over all generator sets $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_1^*, \boldsymbol{g}_2^*, \boldsymbol{g}_3^*\}$ with the required properties, as was found independently by McEliece and Lin [48]. The trellis of any Lee–Lauer code is highly symmetrical. It is fully connected, with 64 branches. Every state can be reached from every other state by a pair of parallel branches corresponding to a coset of the $(8,1,8)$ code in the $(8,7,2)$ code. From the zero state, a branch pair corresponding to the zero coset $\{\mathbf{0}, \mathbf{1}\}$ goes to the zero state, and branch pairs corresponding to complementary weight-4 codewords in the $(8,4,4)$ code go to the seven nonzero states. From any nonzero state, a branch pair corresponding to complementary weight-4 codewords in the $(8,4,4)^*$ code goes to the zero state, while the branch pairs to the seven nonzero states comprise three complementary pairs of weight-4 8-tuples and four complementary pairs of weight-2/weight-6 8-tuples.

[5] According to [1], a Lee–Lauer code was apparently used in the Soviet *Regatta* space-communication system.

From these properties, it follows using straightforward flow-graph techniques that the distance-generating function of any code of the Lee–Lauer type is

$$\begin{aligned}A(z) &= z^8 + \frac{28z^8}{1-4z^2-6z^4-4z^6} \\ &= 29z^8 + 112z^{10} + 616z^{12} + 3248z^{14} + 9968z^{16} + \cdots\end{aligned}$$

Thus the minimum distance is $d = 8$, while $N_8 = 29$, about half that of $\mathbb{C}_{\mathcal{G}}$. However, a Lee–Lauer code has 112 weight-10 code sequences, whereas $\mathbb{C}_{\mathcal{G}}$ has none.

We see that a Lee–Lauer code is even, since all generators have weight 8, but not doubly-even since it is not self-dual. The Lee–Lauer code is also not expressible as a period-4, rate-$1/2$, 16-state code, which hurts its decoding complexity. Using the fast-Hadamard-transform-like techniques of [17] to compute the metrics of all cosets of $(8,1,8)$ in $(8,7,2)$ simultaneously requires 88 additions/subtractions and 64 absolute valuations. The Viterbi algorithm for the resulting fully connected 8-state trellis requires 64 additions and eight eight-way comparisons per 8 bits.

On the other hand, a 16-state rate-$1/2$ code, time-invariant or not, requires only four additions to compute metrics, and then 32 additions and 16 two-way comparisons for Viterbi decoding per unit time (2 bits). The total number of binary decoding operations per 8 bits is therefore 272 for a Lee–Lauer code and only 208 for $\mathbb{C}_{\mathcal{G}}$, even though the former has eight states and the latter 16. The more regular structure of $\mathbb{C}_{\mathcal{G}}$ is also more attractive for implementation. However, the reader should compare this to the very efficient Lee–Lauer decoder developed by Lin and McEliece [43].

ACKNOWLEDGMENT

The authors wish to acknowledge the contributions of many colleagues to this work, particularly Ehud Gelblum, Rolf Johannesson, Ralf Kötter, Frank R. Kschischang, Boris Kudryashov, Hans-Andrea Loeliger, Robert J. McEliece, and Neil J. A. Sloane.

REFERENCES

[1] K. Abdel-Ghaffar, R. J. McEliece, and G. Solomon, "Some partial-unit-memory convolutional codes," Jet Propulsion Lab. TDA Progr. Rep., vol. 42-107, pp. 57–72, Nov. 1991.

[2] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, submitted for publication, July 1998.

[3] O. Amrani, Y. Be'ery, A. Vardy, F.-W. Sun, and H. C. A. van Tilborg, "The Leech lattice and the Golay code: Bounded-distance decoding and multilevel constructions," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1030–1043, July 1994.

[4] J. B. Anderson and S. M. Hladik, "Tail-biting MAP decoders," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 297–302, Feb. 1998.

[5] E. F. Assmus, H. F. Mattson, Jr., and R. J. Turyn, "Research to develop the algebraic theory of codes," Rep. AFCRL-67-0365, Air Force Res. Labs., Bedford, MA, June 1967.

[6] I. E. Bocharova and B. D. Kudryashov, "Rational rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1305–1313, July 1997.

[7] ——, private communication, Sept. 1997.

[8] R. W. D. Booth, M. A. Herro, and G. Solomon, "Convolutional coding techniques for certain quadratic residue codes," in *Proc. Int. Telemetry Conf.* (Silver Springs, MD, 1975).

[9] A. R. Calderbank, G. D. Forney, Jr., and A. Vardy, "Classification of certain tail-biting generators for the binary Golay code," in *Codes,*

Curves, and Signals: Common Threads in Communications, A. Vardy, Ed. Boston, MA: Kluwer, 1998, pp. 127–153.
[10] J. H. Conway, V. S. Pless, and N. J. A. Sloane, "The binary self-dual codes of length up to 32: A revised enumeration," *J. Comb. Theory Ser. A*, vol. 60, pp. 183–195, Jan. 1992.
[11] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 2nd ed. New York: Springer-Verlag, 1993.
[12] M. Cedervall and R. Johannesson, "A fast algorithm for computing the distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1146–1159, Nov. 1989.
[13] E. Gelblum, private communication, Aug. 1997.
[14] J. Feigenbaum, G. D. Forney, Jr., B. H. Marcus, R. J. McEliece, and A. Vardy, Eds., *IEEE Trans. Inform. Theory* (Special Issue on "Codes and Complexity"), vol. 42, Nov. 1996.
[15] G. D. Forney, Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 512–518, July 1973.
[16] ——, "Coset codes—Part I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1123–1151, Sept. 1988.
[17] ——, "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152–1187, Sept. 1988.
[18] ——, "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1741–1752, Nov. 1994.
[19] ——, "Density-length profiles and trellis complexity of lattices," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1753–1772, Nov. 1994.
[20] ——, "The forward-backward algorithm," in *Proc. 34th Annu. Allerton Conf. Communication, Control, and Computing* (Monticello, IL, Oct. 1996), pp. 432–446.
[21] G. D. Forney, Jr., N. J. A. Sloane, and M. D. Trott, "The Nordstrom-Robinson code is the binary image of the octacode," in *Proc. Coding and Quantization* (DIMACS/IEEE Workshop, October 19–21, 1992), A. R. Calderbank *et al.*, Eds. Providence, RI: Amer. Math. Soc., 1993, pp. 19–26.
[22] G. D. Forney, Jr. and M. D. Trott, "The dynamics of group codes: State spaces, trellis diagrams and canonical encoders," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1491–1513, 1993.
[23] B. J. Frey, "Bayesian networks for pattern classification, data compression, and channel coding," Ph.D. dissertation, Univ. Toronto, Toronto, Ont., Canada, July 1997.
[24] A. R. Hammons, Jr., P. V. Kumar, A. R. Calderbank, N. J. A. Sloane, and P. Solé, "The $\mathbb{Z}_4$-linearity of Kerdock, Preparata, Goethals, and related codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 301–319, Mar. 1994.
[25] M. F. Hole and Ø. Ytrehus, "Two-step decoding of partial unit memory convolutional codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 324–330, Jan. 1997.
[26] R. Johannesson, P. Stahl, and E. Wittenmark, private communication, Dec. 1997.
[27] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Codes*. Piscataway, NJ: IEEE Press, 1998.
[28] J. Justesen, "Bounded distance decoding of unit memory codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1616–1627, Sept. 1993.
[29] A. B. Kiely, S. Dolinar, R. J. McEliece, L. Ekroot, and W. Lin, "Trellis decoding complexity of linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1687–1697, Nov. 1996.
[30] R. Kötter and A. Vardy, "Construction of minimal tail-biting trellises," in *Proc. IEEE Inform. Theory Workshop* (Killarney, Ireland, June 1998), pp. 72–74.
[31] ——, "The theory of tail-biting trellises: minimality and basic principles," preprint, May 1999.
[32] ——, "The theory of tail-biting trellises: bounds and applications," manuscript in preparation, May 1999.
[33] ——, private communication, Dec. 1997.
[34] F. R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1924–1937, Nov. 1995.
[35] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 219–230, 1998.
[36] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, submitted for publication, July 1998.
[37] B. D. Kudryashov and T. G. Zakharova, "Block codes from convolutional codes," *Probl. Pered. Inform.*, vol. 25, no. 4, pp. 98–102, Jan. 1989.
[38] A. Lafourcade and A. Vardy, "Asymptotically good codes have infinite trellis complexity," *IEEE Trans. Inform. Theory*, vol. 41, pp. 555–559, Mar. 1995.
[39] ——, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1938–1954, Nov. 1995.
[40] G. S. Lauer, "Some optimal partial-unit-memory codes," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 240–243, Mar. 1979.
[41] L.-N. Lee, "Short unit-memory byte-oriented binary convolutional codes having maximal free distance," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 349–352, May 1976.
[42] P. J. Lee, "There are many good periodically-time-varying convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 460–463, Mar. 1989.
[43] W. Lin and R. J. McEliece, "Hybrid trellis decoding for some convolutional codes," in *Proc. Conf. Information Sciences and Systems* (Princeton, NJ, Mar. 1996), pp. 572–577.
[44] J. H. Ma and J. K. Wolf, "On tail-biting convolutional codes," *IEEE Trans. Commun.*, vol. COM-34, pp. 104–111, Feb. 1986.
[45] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. New York: North-Holland, 1977.
[46] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072–1092, July 1996.
[47] ——, private communication, Jan. 1998.
[48] R. J. McEliece and W. Lin, "The trellis complexity of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1855–1864, Nov. 1996.
[49] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, 1998.
[50] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1049–1053, Sept. 1988.
[51] V. S. Pless, "On the uniqueness of the Golay codes," *J. Combin. Theory*, vol. 5, pp. 215–228, July 1968.
[52] E. M. Rains and N. J. A. Sloane, "Self-dual codes," in *Handbook of Coding Theory*, V.S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, Dec. 1998.
[53] M. Ran and J. Snyders, "A cyclic $[6, 3, 4]$ group code and the hexacode over GF(4)," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1250–1253, July 1996.
[54] A. Reznik, "Iterative decoding of codes defined on graphs," M.Sc. thesis, MIT, Cambridge, MA, May 1998.
[55] S. Riedel and C. Weiss, "The Golay convolutional code and its application," *IEEE Trans. Inform. Theory*, submitted for publication, Jan. 1998.
[56] N. J. A. Sloane, private communication, Aug. 1997.
[57] 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.
[58] M. D. Trott, private communication, Jan. 1998.
[59] A. Vardy, "The Nordstrom-Robinson code: Representation over GF(4) and efficient decoding," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1686–1693, Sept. 1994.
[60] ——, "Even more efficient bounded-distance decoding of the hexacode, the Golay code, and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1495–1499, 1995.
[61] ——, "Trellis structure of codes," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, Dec. 1998.
[62] A. Vardy and Y. Be'ery, "More efficient soft-decision decoding of the Golay codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 667–672, May 1991.
[63] ——, "Maximum-likelihood decoding of the Leech lattice," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1435–1444, July 1993.
[64] A. Vardy and F. R. Kschischang, "Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2027–2034, Nov. 1996.
[65] Q. Wang and V. K. Bhargava, "An efficient maximum-likelihood decoding algorithm for generalized tail-biting codes including quasi-cyclic codes," *IEEE Trans. Commun.*, vol. 37, pp. 875–879, Aug. 1989.
[66] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Comput.*, submitted for publication, July 1998.
[67] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Univ. Linköping, Linköping, Sweden, Apr. 1996.
[68] N. Wiberg, H.-A. Loeliger and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, pp. 513–526, Sept. 1995.
[69] M. Xu and R. J. McEliece, private communication, Sept. 1998.
[70] K. S. Zigangirov and V. V. Chepyzhov, "Study of decoding tail-biting convolutional codes," in *Proc. Swedish–Soviet Workshop Information Theory* (Gotland, Sweden, Aug. 1989), pp. 52–55.