ELSEVIER Theoretical Computer Science 290 (2003) 1775–1797

Theoretical Computer Science

www.elsevier.com/locate/tcs

On viewing block codes as finite automata ☆

Priti Shankar[a,*], Amitava Dasgupta[a], Kaustubh Deshmukh[b], B. Sundar Rajan[c]

[a] *Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India*

[b] *Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA*

[c] *Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India*

Received 15 June 2000; received in revised form 11 September 2001; accepted 14 February 2002
Communicated by M. Nivat

Abstract

Block codes are viewed from a formal language theoretic perspective. It is shown that properties of trellises for subclasses of block codes called rectangular codes follow naturally from the Myhill Nerode theorem. A technique termed subtrellis overlaying is introduced with the object of reducing decoder complexity. Necessary and sufficient conditions for trellis overlaying are derived from the representation of the block code as a group, partitioned into a subgroup and its cosets. The conditions turn out to be simple constraints on coset leaders. It is seen that overlayed trellises are tail-biting trellises for which decoding is generally more efficient than that for conventional trellises. Finally, a decoding algorithm for tail-biting trellises is described, and the results of some simulations are presented.

Keywords: Block codes; Tail-biting trellis; Decoder complexity; Maximum likelihood decoding

1. Introduction

The areas of system theory, coding theory and automata theory have much in common, but historically have developed largely independently. A recent book [13]

☆ The material in this paper was presented in part at STACS 2000, Lille, France.

* Corresponding author. Tel.: +91-812-36441.

E-mail addresses: priti@csa.iisc.ernet.in (P. Shankar), kaustubh@cs.washington.edu (K. Deshmukh), bsrajan@ece.iisc.ernet.in (B.S. Rajan).

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00083-X

elaborates some of the connections. In block coding, an *information sequence* of symbols over a finite alphabet is divided into *message blocks* of fixed length; each message block consists of k information symbols. If q is the size of the finite alphabet, there are a total of q^k distinct messages. Each message is encoded into a distinct *codeword* of n $(n>k)$ symbols. There are thus q^k codewords each of length n and this set forms a *block code* of length n. A block code is typically used to correct errors that occur in transmission over a communication channel. A subclass of block codes, the *linear block codes* has been used extensively for error correction. Traditionally such codes have been described *algebraically*, their algebraic properties playing a key role in *hard decision* decoding algorithms. In hard decision algorithms, the signals received at the output of the channel are *quantized* into one of the q possible transmitted values, and decoding is performed on a block of symbols of length n representing the received codeword, possibly corrupted by some errors. By contrast, *soft decision* decoding algorithms do not require quantization before decoding and are known to provide significant coding gains [4] when compared with hard decision decoding algorithms. That block codes have efficient *combinatorial* descriptions in the form of *trellises* was discovered in 1974 [1]. Two other early papers in this subject were [26,16]. A landmark paper by Forney [6] in 1988 began an active period of research on the trellis structure of block codes. It was realized that the well-known Viterbi Algorithm [23] (which is actually a dynamic programming shortest path algorithm) could be applied to soft decision decoding of block codes. Most studies on the trellis structure of block codes confined their attention to linear codes for which it was shown that unique minimal trellises exist [18]. Trellises have been studied from the viewpoint of linear dynamical systems and also within an algebraic framework [25,7,12,11]. An excellent description of the trellis structure of block codes appears in [22].

The purpose of this paper is to look at trellises for block codes from a formal language theoretic perspective. We first give alternate proofs of some known results on trellises for block codes, which in our view, follow naturally from the Myhill–Nerode theorem [10]. The purpose of viewing the results within this framework is to lay the groundwork for a technique introduced here, termed *subtrellis overlaying*. This essentially splits a single well structured finite automaton representing the code into several smaller automata, which are then *overlayed*, so that they share states. The motivation for this is a reduction in the size of the trellis, in order to improve the efficiency of decoding. We view the block code as a group partitioned into a subgroup and its cosets, and derive necessary and sufficient conditions for overlaying. The conditions turn out to be simple constraints on the coset leaders. The trellises obtained in this manner turn out to belong to the class of *tail-biting* trellises for block codes, recently described in the literature [3]. Section 2 gives a brief review of block codes and trellises; Section 3 views block codes as finite state languages and derives the conditions for overlaying; Section 4 describes the decoding algorithm and presents the results of simulations on a code called the hexacode. Finally Section 5 concludes the paper.

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

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

2. Background

We give a very brief background on subclasses of block codes called linear codes. Readers are referred to the classic texts [17,14,2].

Let F_q be the field with q elements. It is customary to define linear codes algebraically as follows:

Definition 2.1. A *linear block code* C of length n over a field F_q is a k-dimensional subspace of an n-dimensional vector space over the field F_q (such a code is called an (n,k) code).

The most common algebraic representation of a linear block code is the generator matrix G. A $k \times n$ matrix G where the rows of G are linearly independent and which generate the subspace corresponding to C is called a *generator matrix* for C. Fig. 1 shows a generator matrix for a $(4,2)$ linear code over F_2.

A general block code also has a *combinatorial* description in the form of a *trellis*. We borrow from Kschischang et al. [12] the definition of a trellis for a block code.

Definition 2.2. A trellis for a block code C of length n, is an edge labeled directed graph with a distinguished *root* vertex s, having in-degree 0 and a distinguished *goal* vertex f having out-degree 0, with the following properties:

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

The length of a path (in edges) from the root to any vertex is unique and is sometimes called the *time index* of the vertex. One measure of the size of a trellis is the total number of vertices in the trellis. It is well known that minimal trellises for linear block codes are unique [18] and constructable from a generator matrix for the code [12]. Such trellises are known to be *biproper*. Biproperness is the terminology used by coding theorists to specify that the finite state automaton whose transition graph is the trellis, is deterministic, and so is the automaton obtained by reversing all the edges in the trellis. In contrast, minimal trellises for non-linear codes are, in general, neither unique, nor deterministic [12]. Fig. 2 shows a trellis for the linear code in Fig. 1.

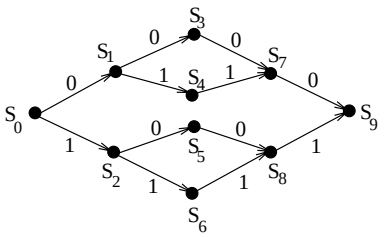


Fig. 2. A trellis for the linear block code of Fig. 1 with $S_0 = s$ and $S_9 = f$.

Willems [25] has given conditions under which an arbitrary block code (which he refers to as a dynamical system) has a unique minimal realization. Forney and Trott [7] have introduced *projections* that split codewords into two parts: a *past* and a *future* with respect to any time index i, $0 \leqslant i \leqslant n$.

Definition 2.3. Let p be any path from the root to a vertex v, and $l(p)$ the label sequence along the path. Then define:

$$P(v) = \{l(p):\ p \text{ is a path from } s \text{ to } v\}$$

$$F(v) = \{l(p):\ p \text{ is a path from } v \text{ to } f\}$$

to be, respectively, the *past* and *future* of v.

Then the code C associated with the trellis is $P(f) = F(s)$. Let V_i be the set of vertices at time index i. Then

$$C = \bigcup_{v \in V_i} P(v)F(v).$$

Thus C can be covered by a union of product form subsets at each time index. This may be visualized by a structure called a *Cartesian Map* similar to a *Karnaugh Map*, where codeword pasts are on one axis and codeword futures on another. The Cartesian Maps in Fig. 3 show past–future relations for the code in Fig. 2 at time indices 1 and 2.

A past–future relation is said to be *rectangular*, if the corresponding Cartesian array can be arranged by row and column permutations into a collection of non-overlapping complete rectangles with no rows or columns in common. The past–future relations depicted in Fig. 3 are thus rectangular.

Codes with the property that past–future relations at each time index are rectangular are called rectangular codes, and properly include the class of linear codes. Such codes are of interest because they correspond exactly to the class of codes that admit biproper trellis representations [11,18]. Biproper trellises minimize a wide variety of structural complexity measures. McEliece [18] has defined a measure of Viterbi decoding

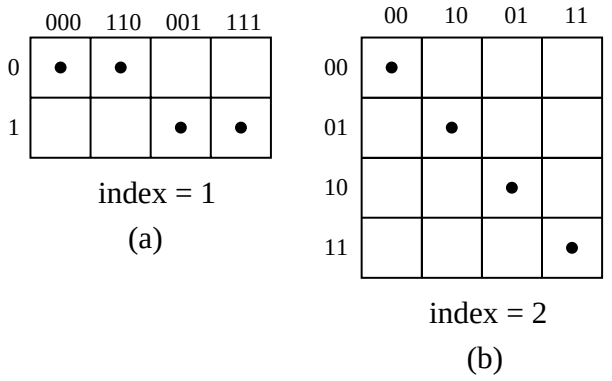


Fig. 3. Past future relationship of a code.

complexity in terms of the number of edges and vertices of a trellis, and has shown that the biproper trellis is the "best" trellis using this measure, as well as other measures based on the maximum number of states at any time index, and the total number of states.

3. Viewing block codes as finite state languages

We begin this section with proofs of equivalences that have been previously established using techniques from linear dynamical systems [25] and algebraic coding theory [18,7,12,11]. These equivalences naturally follow from a modification of the well known Myhill–Nerode theorem [10].

3.1. Bideterministic languages

Let Σ be a finite alphabet, and L be a language over Σ accepted by a finite state automaton. Let L^R be the language obtained by reversing all strings in L. We say that a finite state language is bideterministic if there exists a deterministic finite state automaton (dfa) M accepting L, such that a dfa for L^R is obtained by simply reversing all edges of M. Fig. 4 is an example of a bideterministic language. Fig. 5(a) displays the minimal dfa for a language that is not bideterministic, (though this is not obvious at this point), and Fig. 5(b) displays the minimal dfa for the reverse of the language accepted by the dfa in Fig. 5(a). Bideterministic languages are a subclass of *reversible* languages studied by Pin [19], where it is observed that a language is bideterministic if and only if its minimal automaton is deterministic and codeterministic (the latter meaning that no pair of transitions with the same label enter any state, and the automaton has a unique final state). Kschischang [11], has defined a block code L (which, as we have seen, is a restricted finite state language) to be *rectangular* if it satisfies the following property:

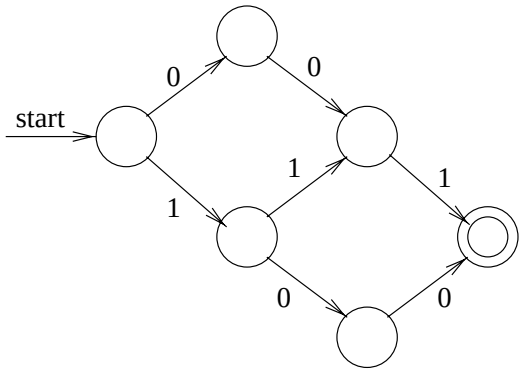


Fig. 4. A deterministic finite state automaton for $L = \{001, 111, 100\}$.

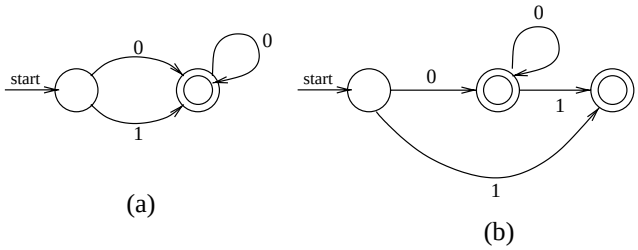


Fig. 5. The minimal deterministic finite state automaton for (a) L represented by the regular expression $(0+1)0^*$ and for (b) the reverse of the language specified in (a).

Property: If x and y are prefixes of L and $(xz, yz, xz') \in L$ then $yz' \in L$ where z and z' are suffices. (In other words, if x and y have a common continuation to a string in L, then they share all continuations to L.)

When rectangularity is defined in this way, it makes sense even for infinite languages including those that are not recognizable by finite state automata. We restrict our attention here to regular languages, and use the term *finite state rectangular* to characterize a rectangular language accepted by a finite state automaton. (It is easy to check that when L is a block code, this description of rectangularity is the same as the geometric description of the previous section.)

Let R be a binary relation on Σ^*. R is said to be *right invariant* whenever $x\ R\ y \Rightarrow xa\ R\ ya$; $x, y \in \Sigma^*,\ \forall a \in \Sigma$. We say that R is *bi-invariant* if in addition to the above property, R also satisfies *right-cancellation* i.e $xa\ R\ ya \Rightarrow x\ R\ y$, $\forall a \in \Sigma$.

We now prove the first theorem which is similar to the Myhill–Nerode theorem (see [10, pp. 65–66]). This theorem is an alternate proof of Theorems 1, 2 and 3 in [11]. We use the quintuple $M = (Q, \Sigma, \delta, q_0, F)$ to define a deterministic finite automaton where Q is the finite set of states, Σ is the input alphabet, δ is the transition function mapping pairs in $Q \times \Sigma$ to Q, q_0 is the initial state and $F \subseteq Q$ is the set of final states.

Theorem 3.1. *The following are equivalent statements*

(i) *L is a bideterministic finite state language over Σ^*.*

(ii) *L is an equivalence class of a bi-invariant equivalence relation over Σ^* of finite index.*

(iii) *Define R_L as follows*: $(x,y)\in R_L$ *iff* $\forall z\in\Sigma^*, [xz\in L$ *whenever* $yz\in L]$; $(x,y)\notin R_L$ *iff* $\forall z\in\Sigma^*\ [xz\in L\Rightarrow yz\notin L]$. *Then R_L is a bi-invariant equivalence relation of finite index with a unique class corresponding to L.*

(iv) *L is a finite state rectangular language.*

Proof. We only give those parts of the proof that are different from the proof in [10] of the Myhill–Nerode theorem.

(i) $\Rightarrow$ (ii): Let $M=(Q,\Sigma,\delta,q_0,F)$ be a reversible dfa accepting L. As in the proof of the Myhill–Nerode theorem, define $R_M\subseteq\Sigma^*\times\Sigma^*$ as follows: $(x,y)\in R_M$ iff $\delta(q_0,x)=\delta(q_0,y)$. We need to show that R_M is bi-invariant. Right invariance follows from the fact that M is a dfa. Also M is bideterministic as it has a unique initial and final state and is reversible. Since every state has at most one incoming transition on a given symbol, $xa\ R_M\ ya\Rightarrow x\ R_M\ y\ \forall a\in\Sigma^*$ and R_M is right-cancellative as well. (We can safely assume that each state of the automaton has a unique error state associated with it, such that all undefined transitions out of that state are assumed to enter the error state. Thus all states will have at most one incoming transition on an input symbol). Then L is the equivalence class associated with the unique final state of M.

(ii) $\Rightarrow$ (iii): Let R be the equivalence relation defined by (ii). Since R is right invariant $(x,y)\in R\Rightarrow(xz,yz)\in R\ \forall z\in\Sigma^*$. Hence $xz\in L\Leftrightarrow yz\in L$. Hence $(x,y)\in R_L$. Assume now that $(x,y)\notin R$. Let z be any string such that $xz\in L$. If yz is also in L, then writing $z=a_1a_2\ldots a_n,\ a_i\in\Sigma$ and using the property of right cancellation repeatedly, we obtain $(x,y)\in R$ contradicting our assumption. Hence $(x,y)\notin R\Rightarrow\forall z$ such that $xz\in L,\ yz\notin L$. Thus $(x,y)\notin R_L$. Hence R is exactly R_L.

(iii) $\Rightarrow$ (iv): We can construct a dfa M_L whose states correspond to the equivalence classes of R_L as in the proof of the Myhill–Nerode theorem. Let $M_L=(Q_L,\Sigma,\delta_L,q_{0L},F_L)$. Denote by $[x]$ the equivalence class containing x, and define $\delta_L([x],a)=[xa]$. $F_L=\{[x]: x\in L\}$. We need to show that L is rectangular. Assume $(xz,yz,xz')\in L$. Since R_L has a unique class corresponding to L (by definition), it follows that $[xz]=[yz]$. By right-cancellation $[x]=[y]$. By right invariance $[xz']=[yz']$ and hence $yz'\in L$, implying L is rectangular.

(iv) $\Rightarrow$ (i): We need to show that L is bideterministic. Let $M=(Q,\Sigma,\delta,q_0,F)$ be the minimal dfa for L. Assume that M is not codeterministic. Then it either has more than one final state or there exists a state q of M and $a\in\Sigma$ such that $\delta(q_1,a)=q,\ \delta(q_2,a)=q$, $q_1\neq q_2$. Let the latter be the case. Let $\delta(q_0,x)=q_1,\ \delta(q_0,y)=q_2$. Since q_1 and q_2 are distinguished in the minimal automaton, there exists a string z such that $xz\in L,\ yz\notin L$. Let $\delta(q,v)\in L$ and $av=w$. We thus have $xw\in L,\ yw\in L,\ xz\in L$ but $yz\notin L$ contradicting the assumption that L is rectangular. Assume that M has more than one final state. Let f_1 and f_2 be final states. Since M is minimal, there exists a distinguishing string, say x such that $\delta(f_1,x)\in F$ but $\delta(f_2,x)\notin F$. Since strings leading to f_1 and f_2 all share one continuation to a string in L namely ε, L cannot be rectangular, as they

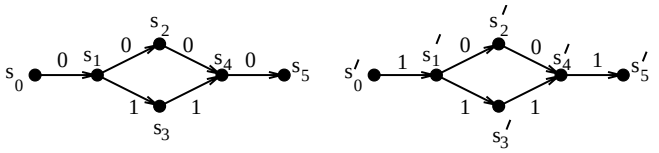


Fig. 6. Minimal trellises for (a) $C_1 = \{0000, 0110\}$ and (b) $C_2 = \{1001, 1111\}$.

do not share all continuations. Since this contradicts our assumption we conclude that L is bideterministic. □

The results above do not require L to be finite. Thus the results for block codes in [11] are a special case of the above theorem.

3.2. Overlaying of subtrellises

We now restrict our attention to linear block codes. As we have mentioned earlier, every linear code has a unique minimal biproper trellis, so this is our starting point. Our object is to describe an operation which we term *subtrellis overlaying*, which yields a smaller trellis. Reduction in the size of a trellis is a step in the direction of reducing decoder complexity.

Let C be a linear (n, k) code with minimal trellis T_C. A *subtrellis* of T_C is a connected subgraph of T_C containing nodes at every time index i, $0 \leqslant i \leqslant n$ and all edges between them. Partition the states of T_C into $n+1$ groups, one for each time index. Let S_i be the set of states corresponding to time index i, and $|S_i|$ denote the cardinality of the set S_i. Define $S_{\max} = \max_i(|S_i|)$. The *state-cardinalityy profile* of the code is defined as the sequence $(|S_0|, |S_1|, \ldots, |S_n|)$. Minimization of $S_{\max}$ is often desirable and $S_{\max}$ is referred to as the *maximum state-cardinality*. Our object here, is to partition the code C into disjoint subcodes, and "overlay" the subtrellises corresponding to these subcodes to get a reduced "shared" trellis. An example will illustrate the procedure.

Example 3.1. Let C be the linear (4,2) code defined by the generator matrix in Fig. 1. C consists of the set of codewords $\{0000, 0110, 1001, 1111\}$ and is described by the minimal trellis in Fig. 2. The state-complexity profile of the code is $(1, 2, 4, 2, 1)$. Now partition C into subcodes C_1 and C_2 as follows:

$$C = C_1 \cup C_2; \quad C_1 = \{0000, 0110\}; \quad C_2 = \{1001, 1111\};$$

with minimal trellises shown in Figs. 6(a) and (b), respectively.

The next step is the "overlaying" of the subtrellises as follows. There are as many states as time index 1 and time index n as partitions of C. States $(s_2, s'_2), (s_3, s'_3), (s_1, s'_1)$, (s_4, s'_4) are superimposed to obtain the trellis in Fig. 7. Note that overlaying may increase the state-cardinality at some time indices (other than 0 and n), and decrease it at others. Codewords are represented by (s_0^i, s_f^i) paths in the overlayed trellis, where

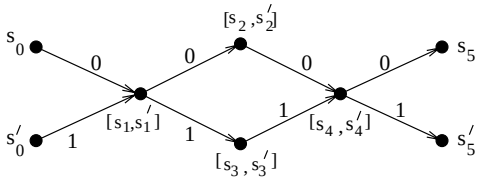


Fig. 7. Trellis obtained by overlaying trellis in Figs. 6(a) and (b).

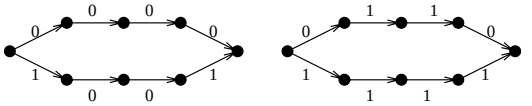


Fig. 8. Minimal subtrellis for (a) $C_1 = \{0000, 1001\}$(b) $C_2 = \{0110, 1111\}$.

s_0^i and s_f^i are the start and final states of subtrellis i. Thus paths from s_0 to s_5 and from s_0' to s_5' represent codewords in the overlayed trellis of Fig. 6. Overlaying forces subtrellises for subcodes to "share" states. Note that the shared trellis is also two way proper, with $S_{\max} = 2$ and state-cardinality profile $(2, 1, 2, 1, 2)$.

Not all partitions of the code permit overlaying to obtain biproper trellises with a reduced value of $S_{\max}$. For instance, consider the following partition of the code.

$$C = C_1 \cup C_2; \quad C_1 = \{0000, 1001\}; \quad C_2 = \{0110, 1111\};$$

with minimal trellis T_1 and T_2 given in Figs. 8(a) and (b), respectively.

It turns out that there exists no overlaying of T_1 and T_2 with a smaller value of $S_{\max}$ than that for the minimal trellis for C.

The small example above illustrates several points. Firstly, it is possible to get a trellis with a smaller number of states to define essentially the same code as the original trellis, with the new trellis having several start and final states, and with a restricted definition of acceptance. Secondly, the new trellis is obtained by the superposition of smaller trellises so that some states are shared. Thirdly, not all decompositions of the original trellis allow superposition to obtain a smaller trellis. The new trellises obtained by this procedure belong to a class termed *tail-biting trellises* described in a recent paper [3]. This class has assumed importance in view of the fact that trellises constructed in this manner can have low state complexity when compared with equivalent conventional trellises. It has been shown [24] that the maximum of the number of states in a tail-biting trellis at any time index could be as low as the square root of the number of states in a conventional trellis at its midpoint. This lower bound however, is not tight, and there are several examples where it is not attained.

Several questions arise in this context. We list two of these below.

1. How does one decide for a given coordinate ordering, whether there exists an overlaying that achieves a given lower bound on the maximum state cardinality at any time index, and in particular, the square root lower bound?

2. Given that there exists an overlaying that achieves a given lower bound how does one find it? That is, how does one decide which states to overlay at each time index?

While, to the best of our knowledge, there are no published algorithms to solve these problems efficiently, in the general case, there are several examples of constructions of minimal tail-biting trellises for specific examples from generator matrices in specific forms in [3], and some recent work on the construction of tail-biting trellises for special subclasses of the linear codes [21,20].

In the next few paragraphs, we define an object called an *overlayed trellis* and examine the conditions under which it can be constructed so that it achieves certain bounds.

Let C be a linear code over a finite alphabet. (Actually a group code would suffice, but all our examples are drawn from the class of linear codes.) Let $C_0, C_1, \ldots, C_l$ be a partition of the code C, such that C_0 is a subgroup of C under the operation of componentwise addition over the structure that defines the alphabet set of the code (usually a field or a ring), and $C_0, C_1, \ldots, C_l$ are cosets of C_0 in C. Let $C_i = C_0 + h_i$ where $h_i, 1 \leqslant h_i \leqslant l$ are coset leaders, with C_i having minimal trellis T_i. C_0 is chosen so that the maximum state complexity is N (occurring at some time index, say, m), where N divides M the maximum state complexity of the conventional trellis at that time index. The codes $C_0, C_1, \ldots, C_l$ are all disjoint subcodes whose union is C. Further, the minimal trellises for $C_0, C_1, \ldots, C_l$ are all structurally identical and two way proper. (That they are structurally identical can be verified by relabeling a path labeled $g_1 g_2 \ldots g_n$ in C_0 with $g_1 + h_{i_1}, g_2 + h_{i_2} \ldots g_n + h_{i_n}$ in the trellis corresponding to $C_0 + h$ where $h = h_{i_1} h_{i_2} \ldots h_{i_n}$.) We therefore refer to $T_1, T_2, \ldots, T_l$ as *copies* of T_0.

Definition 3.1. An *overlayed* biproper trellis is said to exist for C with respect to the partition $C_0, C_1, \ldots, C_l$ where C_i, $0 \leqslant i \leqslant l$ are subcodes as defined above, corresponding to minimal trellises $T_0, T_1, \ldots, T_l$, respectively, with $S_{\max}(T_0) = N$, if it is possible to construct a trellis T_v satisfying the following properties:

1. The trellis T_v has $l+1$ start states labeled $[s_0, \emptyset, \emptyset, \ldots, \emptyset], [\emptyset, s_1, \emptyset \ldots \emptyset] \ldots [\emptyset, \emptyset, \ldots, \emptyset, s_l]$ where s_i is the start state for subtrellis T_i, $0 \leqslant i \leqslant l$.
2. The trellis T_v has $l+1$ final states labeled $[f_0, \emptyset, \emptyset, \ldots, \emptyset], [\emptyset, f_1, \emptyset, \ldots, \emptyset], \ldots [\emptyset, \emptyset, \ldots, \emptyset, f_l]$, where f_i is the final state for subtrellis T_i, $0 \leqslant i \leqslant l$.
3. Each state of T_v has a label of the form $[p_0, p_1, \ldots, p_l]$ where p_i is either $\emptyset$ or a state of T_i, $0 \leqslant i \leqslant l$. Each state of T_i appears in exactly one state of T_v.
4. There is a transition on symbol a from state labeled $[p_0, p_1, \ldots, p_l]$ to $[q_0, q_1, \ldots, q_l]$ in T_v if and only if there is at least one sub-trellis T_i, $0 \leqslant i \leqslant l$ containing a transition from p_i to q_i on symbol a, and for all non-$\emptyset$ components p_j, q_j of the state of T_v, there is a transition from p_j to q_j on input a in T_j. Also every state of T_v has at most one incoming transition on a given symbol and at most one outgoing transition on that symbol.
5. The maximum state cardinality (also called the maximum width) of the trellis T_v at an arbitrary time index i, $1 \leqslant i \leqslant n-1$ is at most N.
6. The set of labels of paths from $[\emptyset, \emptyset, \ldots, s_j, \ldots, \emptyset]$ to $[\emptyset, \emptyset, \ldots, f_j, \ldots, \emptyset]$ is exactly C_j, $0 \leqslant j \leqslant l$.

Let the *state projection* of state $[p_0, p_1, \ldots, p_i, \ldots, p_l]$ into subcode index i be p_i if $p_i \neq \emptyset$ and empty if $p_i = \emptyset$. The *subcode projection* of T_v into subcode index i is defined by the symbol $|T_v|_i$ and consists of the subtrellis of T_v obtained by retaining all the non-$\emptyset$ states in the state projection of the set of states into subcode index i and the edges between them. An overlayed trellis satisfies the property of *projection consistency* which stipulates that $|T_v|_i = T_i$. Thus every subtrellis T_j is embedded in T_v and can be obtained from it by a projection into the appropriate subcode index. We note here that the conventional trellis is equivalent to an overlayed trellis with $M/N = 1$.

To obtain the necessary and sufficient conditions for an overlayed trellis to exist, critical use is made of the fact that C_0 is a group and C_i, $1 \leqslant i \leqslant l$ are its cosets. For simplicity of notation, we denote by G the subcode C_0 and by T, the subtrellis T_0. Assume T has state cardinality profile $(m_0, m_1, \ldots, m_n)$, where $m_r = m_t = N$, and $m_i < N$ for all $i < r$ and $i > t$. Thus r is the first time index at which the trellis attains maximum state cardinality and t is the last time index. Note that it is not necessary that this cardinality be retained between r and t, i.e., the state cardinality may drop between r and t. Since each state of T_v is an M/N-tuple, whose state projections are states in individual subtrellises, it makes sense to talk about a state in T_i corresponding to a state in T_v.

We now give a series of lemmas leading up to the main theorem which gives the necessary and sufficient conditions for an overlayed trellis to exist for a given decomposition of C into a subgroup and its cosets.

Lemma 3.1. *If t is the last time index at which the sub-trellis for the group G attains its maximum state cardinality, any state v of T_v at time index in the range 0 to $t-1$, cannot have more outgoing edges than the corresponding state in T. Similarly, any state v at time index in the range $r+1$ to n in T_v cannot have more incoming edges than the corresponding state in T.*

Proof. Let v be a state at time index i, $i < t$ in T_v. Assume that (v, v') in T_v is an extra outgoing edge. Let T_a be a sub-trellis containing projected states of v and v'. (Properties 3 and 4 guarantee that there is such a sub-trellis.) There will be a path from state v in T_v to a state, say v_t of T_v at time index t beginning with the extra edge. Every state in T_v at time index t has a non-$\emptyset$ projection to a state in T_a as both, the overlayed trellis, as well as the sub-trellis have the same number of states at that time index. Hence there is a path from the projection of v_t into T_a to the terminal vertex of T_a. This either gives an extra codeword in the coset represented by T_a as there is an extra edge out of the projection of v in T_a, or an additional path for an existing codeword, both of which are not possible. The case $i > r$ can be handled similarly. Hence the lemma. □

We say that subtrellises T_a and T_b *share* a state v of T_v at level i if v has non-$\emptyset$ state projections in both T_a and T_b at time index i.

Lemma 3.2. *If the trellises T_a and T_b share a state, say v at level $i \leqslant t$ then they share states at all levels j such that $i \leqslant j \leqslant t$. Similarly, if they share a state v at level $i \geqslant r$, then they share states at all levels j such that $r \leqslant j \leqslant i$.*

Proof. Let v at time index i be a vertex shared between T_a and T_b. Assume there exists a path in T_v from v to some state, say v_t at time index t. By Lemma 3.1, all states on this path cannot have more outgoing edges than the projected states in T_a and T_b. Thus any edge in the path from v to v_t must have corresponding edges in both T_a as well as T_b. Since both v and v_t have non-$\emptyset$ projections in T_a and T_b, the whole path is in T_a as well as T_b. Hence T_a and T_b share states at all levels j such that $i \leqslant j \leqslant t$. The other case can be proved in a similar manner. □

Lemma 3.3. *If trellises T_a and T_b share a state at time index i, then they share all states at time index i.*

Proof. *Case* $i = t$: Obvious.

Case $i < t$: From Lemma 3.2 there is a path from level i to level t which is common to both T_a and T_b. Let h_a and h_b be codewords in T_a and T_b, respectively, which contain this path as a subpath. Hence, T_a is the trellis for the coset $C_a = G + h_a$ and T_b the trellis for $C_b = G + h_b$. Hence T_b is the trellis for the coset $C_a - h_a + h_b$. Since h_a and h_b are identical from time index i to t, $-h_a + h_b$ is 0 from time indices i to t. Thus trellises T_a and T_b are the same from levels i to t. Hence T_a and T_b share all states at level i.

Case $i > t$: This case is similar to the case $i < t$. □

Lemma 3.4. *If T_a and T_b share states at levels $i-1$ and i, then their coset leaders have the same symbol at level i.*

Proof. Follows from Lemma 3.3.

We use the following terminology. If h is a codeword say $h_1 h_2 \ldots h_n$, then for $i < t$, $h_{i+1} \ldots h_t$ is called the *tail* of h at i; for $i > r$, $h_r \ldots h_i$ is called the *head* of h at level i.

Lemma 3.5. *If T_a and T_b have common states at level $i < t$, then there exist coset leaders h_a and h_b of the cosets corresponding to T_a and T_b such that h_a and h_b have the same* tails *at level i. Similarly, if $i > r$ there exist h_a and h_b such that they have the same heads at level i.*

Proof. Follows from Lemmas 3.2 and 3.4.

Now each of the M/N copies of T has m_i states at level i, $0 \leqslant i \leqslant n$. Since the width of the overlayed trellis cannot exceed N for $1 \leqslant i \leqslant n-1$, at least $(M/N^2) \times m_i$ copies of trellis T must be overlayed at time index i. Thus there are at most N/m_i (i.e. $(M/N)/((M/N^2 \times m_i))$) groups of trellises that are overlayed on one another at time index i. From Lemma 3.5 we know that if S is a set of trellises that are overlayed on one another at level i, $i < t$, then the coset leaders corresponding to these trellises have the same tails at level i. Similarly, if $i > r$ the coset leaders have the same heads

at level i. This leads us to the main theorem which we will henceforth refer to as the *overlaying theorem.*

Theorem 3.2. *Let G be a subgroup of the group code C under componentwise addition over the appropriate structure, with $S_{\max}(T_C)=M$, $S_{\max}(T)=N$ and let G have M/N cosets with coset leaders $h_0, h_1, \ldots, h_{M/N-1}$. Let t, r be the time indices defined earlier. Then C has an overlayed proper trellis T_v with respect to the cosets of G if and only if*:

For all i in the range $1 \leqslant i \leqslant n-1$ there exist at most N/m_i collections of coset leaders such that

(i) *If $1 \leqslant i < t$, then the coset leaders within a collection have the same tails at level i.*
(ii) *If $r < i < n$, the coset leaders within a collection have the same heads at level i.*

Proof. (only if) This follows from the argument preceding the statement of the theorem in Section 3. (if) Assume that coset leaders satisfying the conditions above exist. Then at each level, we overlay exactly those trellises whose coset leaders lie in the same collection. Each time index i, $1 \leqslant i \leqslant n-1$ of T_v will have at most $N/m_i \times m_i$ states. Since the tails are identical, the trellises from levels i to t are identical, and so also for the heads. Hence the overlaying is possible. □

Corollary 3.1. *If $M = N^2$ and the conditions of the theorem are satisfied, we obtain a trellis which satisfies the square root lower bound.*

The proof of Theorem 3.2 and Corollary 3.1 answer both the questions about overlayed trellises posed earlier. However the problem of the existence of an efficient algorithm for the decomposition of a code into an appropriate subgroup and its cosets remains open. A non-trivial example illustrating an overlayed trellis is displayed below.

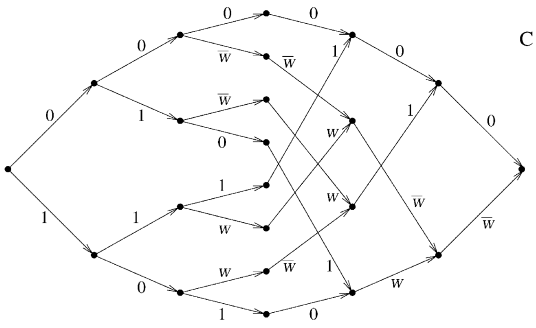
3.3. Example illustrating an overlayed trellis

Example 3.2. We now present an example of an overlayed trellis for a non-trivial code called the hexacode. The hexacode H_6 is a $(6,3)$ code over the quaternary field $\mathbf{F}_4 = \{0, 1, \omega, \bar{\omega}\}$ [5]. The following matrix is a generator matrix for the hexacode

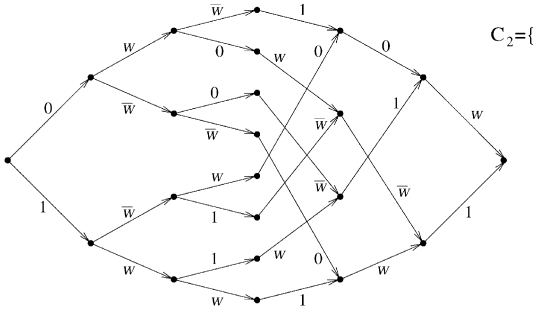
$$G = \begin{bmatrix} 11 & 11 & 00 \\ 00 & 11 & 11 \\ 10 & 01 & \omega\bar{\omega} \end{bmatrix}.$$

The minimal trellis for this code has state-complexity profile $(1,4,16,64,16,4,1)$.

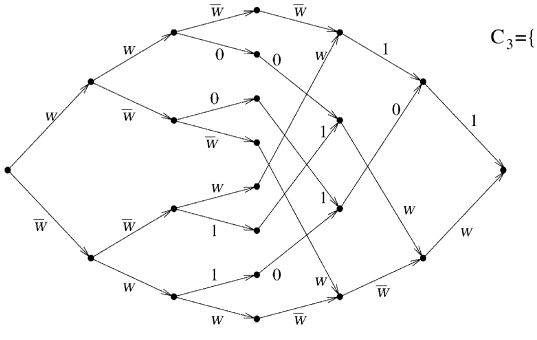
The hexacode has total of $4^3 = 64$ codewords. A tail-biting trellis has been constructed for the hexacode in [3] with state-complexity profile (8,8,8,8,8,8,8). We now construct an overlayed trellis for H_6 which is identical to the taibiting trellis of [3]. Figs. 9 and 10 indicate the eight subcodes $C_1, C_2, \ldots, C_8$ of H_6 and their corresponding minimal biproper trellises. The overlayed trellis is displayed in Fig. 11.



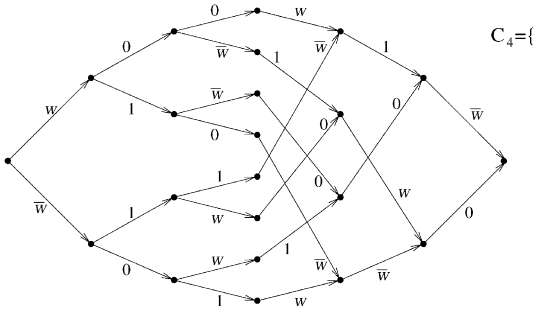
$C_1 = \{\ 000000\ \ 111100$
$00\overline{w}w\overline{w}w\ \ 11w\overline{w}w\overline{w}$
$1010w\overline{w}\ \ 0101w\overline{w}$
$10w\overline{w}10\ \ 01\overline{w}w10\ \}$



$C_2 = \{\ 0w\overline{w}10w\ \ 1\overline{w}w00w$
$0w0w\overline{w}1\ \ 1\overline{w}1\overline{w}w1$
$1ww1w1\ \ 0\overline{w}\overline{w}0w1$
$1w1w1w\ \ 0\overline{w}0\overline{w}1w\ \}$

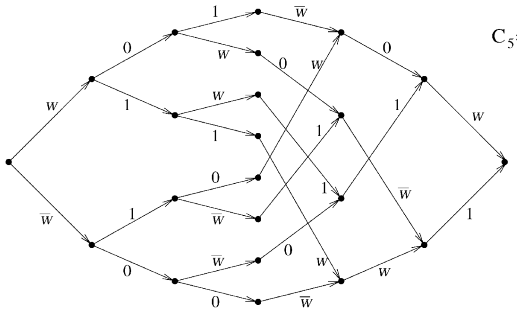


$C_3 = \{\ ww\overline{w}w11\ \ \overline{w}\overline{w}ww11$
$ww00ww\ \ \overline{w}\overline{w}11ww$
$\overline{w}ww\overline{w}\overline{w}w\ \ w\overline{w}\overline{w}w\overline{w}\overline{w}w$
$\overline{w}w1001\ \ w\overline{w}0101\ \}$

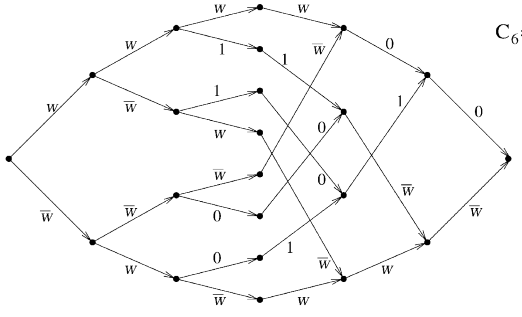


$C_4 = \{\ w00w1\overline{w}\ \ \overline{w}11\overline{w}1\overline{w}$
$w0\overline{w}1w0\ \ \overline{w}1w0w0$
$\overline{w}01w\overline{w}0\ \ w10\overline{w}\overline{w}0$
$\overline{w}0w10\overline{w}\ \ w1\overline{w}00\overline{w}\ \}$

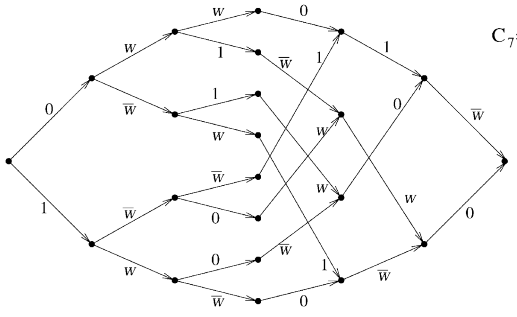
Fig. 9. Subtrellises for the subcodes C_1, C_2, C_3 and C_4 of H_6.



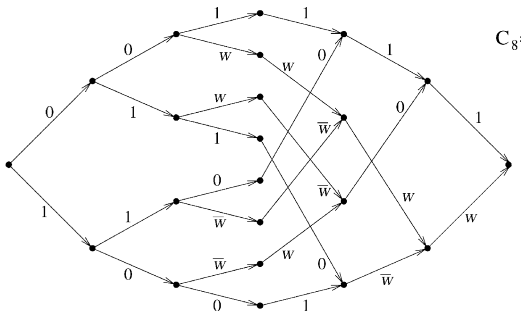
C_5={ $w01\overline{w}0w$ $\overline{w}10w0w$
$w0w0\overline{w}1$ $\overline{w}1\overline{w}1\overline{w}1$
$\overline{w}00\overline{w}w1$ $w11ww1$
$\overline{w}0\overline{w}01w$ $w1w11w$ }



C_6={ $wwww00$ $\overline{wwww}00$
$ww11\overline{ww}$ $\overline{ww}00\overline{ww}$
$\overline{w}w\overline{w}w\overline{w}w$ $w\overline{w}w\overline{w}w\overline{w}$
$\overline{w}w0110$ $w\overline{w}1010$ }



C_7={ $0ww01\overline{w}$ $1\overline{ww}11\overline{w}$
$0w1\overline{w}w0$ $1\overline{w}0ww0$
$1w\overline{w}0\overline{w}0$ $0\overline{w}w1\overline{w}0$
$1w0\overline{w}0\overline{w}$ $0\overline{w}1w0\overline{w}$ }



C_8={ 001111 110011
$00wwww$ $11\overline{ww}ww$
$1001\overline{w}w$ $0110\overline{w}w$
$10\overline{w}w01$ $01w\overline{w}01$ }

Fig. 10. Subtrellises for the subcodes C_5, C_6, C_7 and C_8 of H_6.

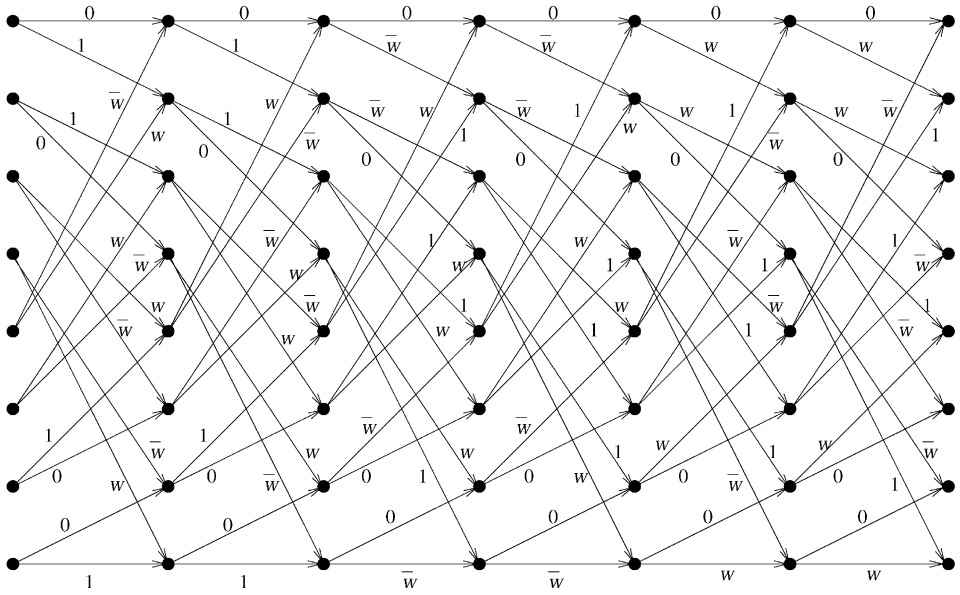


Fig. 11. The overlayed trellis for H_6.

Choose coset leaders $h_0, h_1, \ldots, h_7$ as follows:

$$h_0 = (0,0,0,0,0,0),$$

$$h_1 = (1,\bar{\omega},\omega,0,0,\omega),$$

$$h_2 = (\omega,\omega,0,0,\omega,\omega),$$

$$h_3 = (\bar{\omega},1,\omega,0,\omega,0),$$

$$h_4 = (\bar{\omega},0,0,\bar{\omega},\omega,1),$$

$$h_5 = (\omega,\bar{\omega},\omega,\bar{\omega},\omega,\bar{\omega}),$$

$$h_6 = (1,\omega,0,\bar{\omega},0,\bar{\omega}),$$

$$h_7 = (0,1,\omega,\bar{\omega},0,1).$$

Trellis T will be a member of an overlayed collection consisting of two trellises at time index 1, four trellises at time index 2, eight trellises at time index 3, four trellises at time index 4, two trellises at time index 5, and one at time index 6. Identifying each trellis with its coset with $G = h_0$, the list below consists of trellises in the collection of which G is a member. The list consists of (time index, set of coset leaders) pairs.

$(0,\{h_0\}),(1,\{h_0,h_4\}),(2,\{h_0,h_2,h_4,h_6\})$, $(3,\{h_0,h_1,h_2,h_3,h_4,h_5,h_6,h_7\})$, $(4,\{h_0,h_1,h_2,h_3\}),(5,\{h_0,h_1\}),(6,\{h_0\})$.

3.4. Alternate formulation of the overlaying condition

We now formulate the conditions for overlaying automata represented by trellises, in terms of the equivalence relations of the Myhill–Nerode theorem. We define two sequences of equivalence relations on the set of equivalence relations of the Myhill–Nerode theorem for each subcode and show that the overlaying condition is just a condition on the indices of these sequences of equivalence relations.

Let $R_1,R_2,\ldots,R_l$ be the equivalence relations of the Myhill–Nerode theorem for the subcodes $C_0,C_1,\ldots,C_l$, respectively. Let $\mathscr{R}$ be the set $\{R_1,R_2,\ldots,R_l\}$. We define sequences of equivalence relations $\mathscr{S}_{it}$, $0\leqslant i<t$ and $\mathscr{Q}_{rj}$, $r<j\leqslant n$ as follows:

$\mathscr{S}_{it}\subseteq\mathscr{R}\times\mathscr{R}$. $(R_a,R_b)\in\mathscr{S}_{it}$ if[for x,y,x',y', prefixes of C with lengths i, $1\leqslant i\leqslant t$ and for all $z\in F_q^*$, with $1\leqslant length(z)\leqslant(t-i)$, such that $xz,yz,x'z,y'z$ are prefixes of C, (xR_ay) and $(x'R_by')\Leftrightarrow(xzR_ayz)$ and $(x'zR_by'z)$]

$\mathscr{Q}_{rj}\subseteq\mathscr{R}\times\mathscr{R}$. $(R_a,R_b)\in\mathscr{Q}_{rj}$ if for x,y,x',y' prefixes of C with lengths j, $r\leqslant j\leqslant n-1$, for all $z\in F_q^*$, with $1\leqslant length(z)\leqslant((n-1)-j)$, and such that $xz,yz,x'z,y'z$ are prefixes of C, (xzR_ayz) and $(x'zR_by'z)\Leftrightarrow(xR_ay)$ and $(x'R_bz)$.

The overlaying theorem can be reformulated in terms of equivalence relations as follows:

Theorem 3.3. *Let G be a subgroup of the code C under componentwise addition over the appropriate structure, with $S_{\max}(T_C)=M$ and $S_{\max}(T)=N$, with G having $M/N=l$ cosets, $C_0=G,C_1,\ldots,C_l$. Let t,r be the time indices defined earlier. Then C has an overlayed biproper trellis T_v with respect to the subcodes $C_0,C_1,\ldots,C_l$ if and only if*

1. *The index of $\mathscr{S}_{it}$ is at most N/m_i, $1\leqslant i\leqslant t$.*
2. *The index of $\mathscr{Q}_{rj}$ is at most N/m_i, $r\leqslant j\leqslant(n-1)$,*
   where $(m_0,m_1,m_2,\ldots,m_n)$ is the (length, state—complexity) profile of the trellis T for G.

Proof. That the theorem is true follows from the following observations:

1. The states for the subtrellis of each subcode C_a are identified with the equivalence classes of equivalence relation R_a
2. Two equivalence relations R_a and R_b are put into the same class of $\mathscr{S}_{it}$ if there exist coset leaders for C_a and C_b which have the same tails at level i.
3. Two equivalence relations R_a and R_b are put into the same class of $\mathscr{Q}_{rj}$ if there exist coset leaders for the subgroups C_a and C_b which have the same heads at level j.
4. The condition on the index of the equivalence relations follows from the overlaying theorem. □

4. Decoding

Decoding refers to the process of forming an estimate of the transmitted codeword **x** from a possibly garbled received version **y**. The received vector **y** consists of a sequence of n real numbers, where n is the length of the code. Maximum likelihood soft decision decoding, implies the minimization of the probability of decoding to an incorrect codeword when all the codewords have equal probability of being transmitted. The soft decision decoding algorithm can be viewed as a shortest path algorithm on the trellis for the code. Based on the received vector, a cost $l(u,v)$ can be associated with an edge from node u to node v. The well-known Viterbi decoding algorithm [23] is essentially a dynamic programming algorithm, used to compute a shortest path from the source to the goal node.

4.1. The Viterbi decoding algorithm

For purposes of this discussion, we assume that the cost is a non-negative number. Since the trellis is a regular layered graph, the algorithm proceeds level by level, computing a *survivor* at each node; this is a shortest path to the node from the source. For each branch b, leaving a node at level i, the algorithm updates the survivor at that node by adding the cost of the branch to the value of the survivor. For each node at level $i+1$, it compares the values of the path cost for each branch entering the node and chooses the one with minimum value. There will thus be only one survivor at the goal vertex, and this corresponds to the decoded codeword. For an overlayed trellis we are interested only in paths that go from s_i to f_i, $0 \leqslant i \leqslant l$.

4.2. The A^* algorithm

The A^* algorithm is well known in the literature on artificial intelligence [9] and is a modification of the Dijkstra shortest path algorithm. That the A^* algorithm can be used for decoding was demonstrated in [8]. The A^* algorithm uses, in addition to the path length from the source to the node u, an estimate $h(u,f)$ of the shortest path length from the node u to the goal node in guiding the search. Let $L_T(u,f)$ be the shortest path length from u to f in T. Let $h(u,f)$ be any lower bound such that $h(u,f) \leqslant L_T(u,f)$, and such that $h(u,f)$ satisfies the following inequality, i.e., for u a predecessor of v, $l(u,v) + h(v,f) \geqslant h(u,f)$. If both the above conditions are satisfied, then the algorithm A^*, on termination, is guaranteed to output a shortest path from s to f. The algorithm is given below.

Algorithm A^*.

Input: A trellis $T=(V,E,l)$ where V is the set of vertices, E is the set of edges and $l(u,v) \geqslant 0$ for edge (u,v) in E, a source vertex s and a destination vertex f, and an estimate $h(u,f)$ for the shortest path from u to f for each vertex $u \in V$.

Output: The shortest path from s to f.

/* $p(u)$ is the cost of the current shortest path from s to u and $P(u)$ is a current shortest path from s to u */

begin
$S \leftarrow \emptyset$, $\bar{S} \leftarrow \{s\}$, $p(s) \leftarrow 0$, $P(u) \leftarrow ()$, $\forall u \in V$, $p(u) = +\infty, \forall u \neq s$;
repeat
Let u be the vertex in $\bar{S}$ with minimum value of $p(u) + h(u, f)$.
$S \leftarrow S \cup \{u\}$; $\bar{S} \leftarrow \bar{S} \backslash \{u\}$;
if $u = f$ *then return* $P(f)$;
for each $(u, v) \in E$ *do*
if $v \notin S$ *then*
begin
$p(v) \leftarrow \min(p(u) + l(u, v), previous(p(v)))$;
if $p(v) \neq previous(p(v))$ *then* append (u, v) to $P(u)$ to give $P(v)$;
$(\bar{S}) \leftarrow (\bar{S}) \cup \{v\}$;
end
forever
end

4.3. Decoding on an overlayed trellis

We first define the notation we will be using. A path corresponding to a *codeword* in the tail-biting trellis will be denoted by an $s_i - f_i$ path. A path corresponding to a *non-codeword* is denoted by an $s_i - f_j$ path $i \neq j$. The estimate or lower bound for a shortest path from a node u in a sub-trellis to the final node f_j in that sub-trellis is denoted by $h(u, f_j)$. The l sub-trellises are denoted by $T_1, T_2, \ldots, T_l$, with sub-trellis T_j corresponding to sub-code C_j. A Viterbi algorithm on the overlayed trellis is just a dynamic programming algorithm that finds shortest paths to the destination nodes $f_1, f_2, \ldots, f_l$ from *any* of the source nodes $s_1, s_2, \ldots, s_l$. A *survivor* at any intermediate node u is a shortest path from a source node s_i to u. A *winning* path or a *winner* at node f_j is a survivor at node f_j and its cost is denoted by $h(s_j, f_j)$, as it turns out to be a lower bound on the cost of a shortest path from s_j to f_j. (Note that such a path need not start at s_j.)

Decoding on an overlayed trellis needs at most two phases. In the first phase, a Viterbi algorithm is run on the overlayed trellis T_v with $l+1$ start states and $l+1$ final states. The aim of this phase is to obtain survivors at each node, which will subsequently be used to obtain estimates $h(u, f_j)$ for nodes u in trellis T_j. These estimates are used in the A^* algorithm that is run on subtrellises in the second phase. The winner in the first phase is either an $s_j - f_j$ path, in which case the second phase is not required, or an $s_i - f_j$ path, $i \neq j$, in which case the second phase is necessary. During the second phase, decoding is performed on one subtrellis at a time, the *current* subtrellis, say T_j (corresponding to subcode C_j) being presently the most promising one, in its potential to deliver the shortest path. If at any point, the computed estimate of the shortest path in the current subtrellis exceeds the minimum estimate among the rest of the subtrellises, currently held by, say, subtrellis T_k, then the decoder switches from T_j to T_k, making

T_k the current subtrellis. Decoding is complete when a final node is reached in the current subtrellis. To summarize, either the first phase delivers the winner, i.e. the shortest $s_j - f_j$ path, or the second phase does, in the latter case, the winner being the first path in the current subtrellis to reach its final node. The two phases are described below. (All assertions in italics are proved in the next subsection).

Phase 1: Execute a Viterbi decoding algorithm on the shared trellis, and obtain survivors at each node. *Each survivor at a node u has a cost which is a lower bound on the cost of the least cost path from s_j to u in an $s_j - f_j$ path passing through u, $1 \leqslant j \leqslant N$. If there exists a value of k for which an $s_k - f_k$ path is an overall winner then this is the shortest path in the original trellis T_C.* If this happens decoding is complete. If no such $s_k - f_k$ path exists go to Phase 2.

Phase 2: 1. Consider only subtrellises T_j such that the winning path at T_j is an $s_i - f_j$ path with $i \neq j$ (i.e at some intermediate node a prefix of the $s_j - f_j$ path was "knocked out" by a shorter path originating at s_i), and such that there is no $s_k - f_k$ path with smaller cost. Let us call such trellises *residual trellises*. Let the estimate $h(s_j, f_j)$ associated with the node s_j be the cost of the survivor at f_j obtained in the first phase.

2. Create a heap of r elements where r is the number of residual trellises, with current estimate $h(s_j, f_j)$ with minimum value as the top element. Let j be the index of the subtrellis with the minimum value of the estimate. Remove the minimum element corresponding to T_j from the heap and run the A^* algorithm on trellis T_j (called the *current trellis*). For a node u, take $h(u, f_j)$ to be $h(s_j, f_j) - cost(survivor(u))$ where $cost(survivor(u))$ is the cost of the survivor obtained in the first phase. The quantity $h(u, f_j)$ *satisfies the two properties required of the estimator in the A^* algorithm.*

3. At each step, compare $p(u) + h(u, f_j)$ in the current subtrellis with the top value in the heap. If at any step the former exceeds the latter (associated with subtrellis, say, T_k), then make T_k the current subtrellis. Insert the current value of $p(u) + h(u, f_j)$ in the heap (after deleting the minimum element) and run the A^* algorithm on T_k either from start node s_k (if T_k was not visited earlier) or from the node which it last expanded in T_k. Stop when the goal vertex is reached in the current subtrellis.

In the best case (if the algorithm needs to execute Phase 2 at all) the search will be restricted to a single residual subtrellis; the worst case will involve searching through all residual subtrellises.

4.4. Proofs of assertions in the decoding algorithm

Lemma 4.1. *Each survivor at a node u has a cost which is a lower bound on the cost of the least cost path from s_j to u in an $s_j - f_j$ path passing through u.*

Proof. Assume that u is an arbitrary node on an $s_j - f_j$ path and that path P is the survivor at u in the first phase. There are two cases. Either P is a path from s_j to u or P is a path from s_i to u, $j \neq i$. If the latter is the case, then the cost of P is less than the cost of the path from s_j to u; hence the cost of the survivor at u is a lower bound on the cost of the least cost path from s_j to u. □

Lemma 4.2. *If there exists a value of k for which an $s_k - f_k$ path is an overall winner in the overlayed trellis T_v, then this is the shortest path from source to goal in the original trellis T_C.*

Proof. The set of paths in T_v from time index 0 to time index n is a superset of paths from the source to the goal vertex in T_C, as only $s_k - f_k$ paths in T_v correspond to paths in T_C. Thus if an $s_k - f_k$ path is an overall winner in T_v, it must be a winner in T_C as well. □

Lemma 4.3. *The quantity $h(u, f_j)$ defined in the algorithm satisfies the following two properties*:
1. $h(u, f_j) \leqslant L_{T_j}(u, f_j)$,
2. $l(u,v) + h(v, f_j) \geqslant h(u, f_j)$ *where* (u,v) *is an edge.*

Proof. 1. $h(u, f_j) = cost(survivor(f_j)) - cost(survivor(u))$.
Also $cost(survivor(f_j)) \leqslant cost(survivor(u)) + L_{T_j}(u, f_j)$, from which the result follows.
2. To prove: $l(u,v) + h(v, f_j) \geqslant h(u, f_j)$

$$\begin{aligned} \text{LHS} &= l(u,v) + h(v, f_j) \\ &= l(u,v) + h(s_j, f_j) - cost(survivor(v)). \end{aligned}$$

If survivor at v is survivor at u concatenated with edge (u,v), then

$$\begin{aligned} \text{LHS} &= l(u,v) + h(s_j, f_j) - cost(survivor(u)) - l(u,v) \\ &= h(u, f_j). \end{aligned}$$

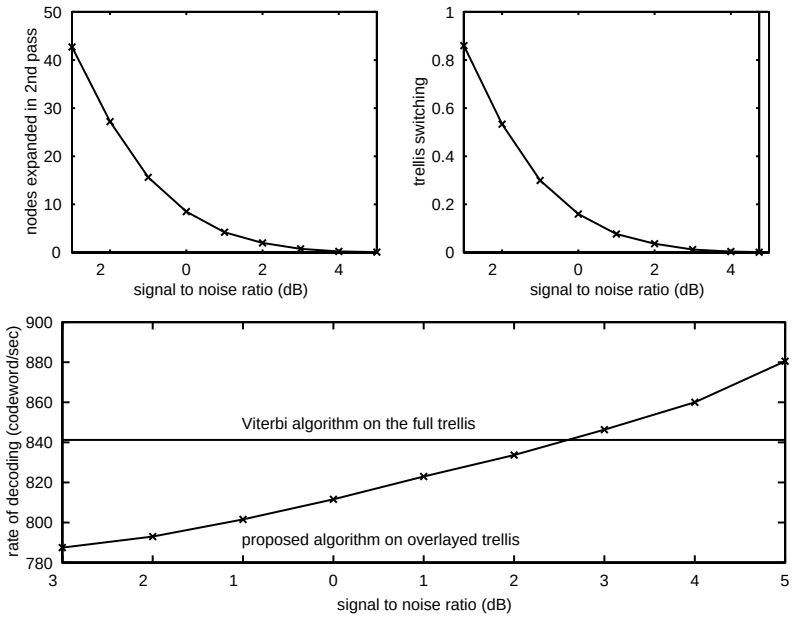
On the other hand if survivor at v is not a continuation of the survivor at u,

$$\begin{gathered} cost(survivor(v)) < cost(survivor(u)) + l(u,v) \\ cost(survivor(v)) - l(u,v) < cost(survivor(u)) \\ \text{or, } h(s_j, f_j) - cost(survivor(v)) + l(u,v) > h(s_j, f_j) - cost(survivor(u)) \\ \text{or, } h(v, f_j) + l(u,v) > h(u, f_j) \\ \text{Therefore, } l(u,v) + h(v, f_j) \geqslant h(u, f_j). \quad \square \end{gathered}$$

4.5. Simulation results

Below we display the results of simulations using, both the Viterbi decoding algorithm on the original trellis, and the algorithm of Section 4.3 on the overlayed trellis. There are a total of 56 nodes in the graph. The graph showing the average number of nodes expanded in the second pass, indicates that the second pass is rarely needed for signal-to-noise ratio of 2 dB or greater. The second graph shows how many times decoding begins on a new subtrellis (normalized on a scale of ten.) The third

graph shows that the proposed algorithm outperforms the Viterbi algorithm on the conventional trellis for signal to noise ratios of about 2.5 dB and beyond.



5. Conclusions

This paper offers a new perspective from which block codes may be fruitfully viewed. It is shown that previous results on minimal trellises for rectangular block codes follow naturally from the Myhill–Nerode theorem, when the trellis is viewed as a finite automaton. A technique called overlaying is proposed here which reduces the size of the trellis and produces a tail-biting trellis. Necessary and sufficient conditions for overlaying are derived from the representation of the code as a group. Finally a decoding algorithm is proposed which needs at most two passes on the tail-biting trellis. The algorithm is a general algorithm which will work on any tail-biting trellis. Simulation results suggest that overlaying may offer significant benefits on larger trellises. Future work will concentrate on investigating the existence of an efficient algorithm for finding tail-biting trellises and on obtaining tail-biting trellises for industrial strength codes.

Acknowledgements

The authors are very grateful to the referee for his careful reading of the manuscript and his useful comments which improved the presentation of the paper.

References

[1] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, IEEE Trans. Inform. Theory 20 (2) (1974) 284–287.
[2] R.E. Blahut, Theory and Practice of Error Control Codes, Addison Wesley, Reading, MA, 1984.
[3] A.R. Calderbank, G. David Forney Jr., A. Vardy, Minimal tail-biting trellises: the Golay code and more, IEEE Trans. Inform. Theory 45 (5) (1999) 1435–1455.
[4] G.C. Clark Jr., J.B. Cain, Error-Correction Coding for Digital Communication, Plenum Press, New York, 1981.
[5] J.H. Conway, N.J.A. Sloane, Sphere Packings, Lattices and Groups, Springer, New York, 1993.
[6] G.D. Forney Jr., Coset codes II: binary lattices and related codes, IEEE Trans. Inform. Theory 36 (5) (1988) 1152–1187.
[7] G.D. Forney Jr., M.D. Trott, The dynamics of group codes: state spaces, trellis diagrams and canonical encoders, IEEE Trans. Inform. Theory 39 (5) (1993) 1491–1513.
[8] Y.S. Han, C.R.P. Hartmann, C.C. Chen, Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes, IEEE Trans. Inform. Theory 39 (5) (1993) 714–729.
[9] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Solid-State Circuits SSC-4 (1968) 100–107.
[10] J.E. Hopcroft, J.D. Ullman, Introduction to Automata, Languages, and Computation, Addison Wesley, Reading, MA, 1977.
[11] F.R. Kschischang, The trellis structure of maximal fixed cost codes, IEEE Trans. Inform. Theory 42 (6) (1996) 1828–1837.
[12] F.R. Kschischang, V. Sorokine, On the trellis structure of block codes, IEEE Trans. Inform. Theory 41 (6) (1995) 1924–1937.
[13] D. Lind, M. Marcus, An Introduction to Symbolic Dynamics and Coding, Cambridge University Press, Cambridge, 1995.
[14] F.J. MacWilliams, N.J.A. Sloane, The Theory of Error Correcting Codes, North-Holland, Amsterdam, 1981.
[15] J.L. Massey, Foundations and methods of channel encoding, in: Proc. Int. Conf. on Information Theory and Systems, Vol. 65, Berlin, Germany, 1978.
[16] R.J. McEliece, The Theory of Information and Coding, Encyclopedia of Mathematics and its Applications, Addison Wesley, Reading, MA, 1977.
[17] R.J. McEliece, On the BCJR trellis for linear block codes, IEEE Trans. Inform. Theory 42 (1996) 1072–1092.
[18] D.J. Muder, Minimal trellises for block codes, IEEE Trans. Inform. Theory 34 (5) (1988) 1049–1053.
[19] J.E. Pin, On reversible automata, in: I. Simon (Ed.), LATIN 92, Lecture Notes in Computer Science, Vol. 583, Springer, Berlin, 1992, pp. 401–416.
[20] P. Shankar, P.N.A. Kumar, H. Singh, B.S. Rajan, Minimal Tail-Biting Trellises for certain cyclic block codes are easy to construct, in: F. Orejas, P. Spirakis, J. van Leeuwen (Eds.), ICALP 2001, Lecture Notes in Computer Science, Vol. 2076, Springer, Berlin, 2001, pp. 627–638.
[21] H. Singh, On tail-biting trellises for linear block codes, M.E. Thesis, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore-560012, January 2001.
[22] A. Vardy, Trellis structure of codes, in: V.S. Pless, W.C. Huffman (Eds.), Handbook of Coding Theory, Elsevier Science, Amsterdam, 1998.
[23] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inform. Theory 13 (1967) 260–269.
[24] N. Wiberg, H.-A. Loeliger, R. Kotter, Codes and iterative decoding on general graphs, Eoro. Trans. Telecommun. 6 (1995) 513–526.
[25] J.C. Willems, Models for dynamics, in: U. Kirchgraber, H.O. Walther (Eds.), Dynamics Reported, Vol. 2, Wiley, New York, 1989, pp. 171–269.
[26] J.K. Wolf, Efficient maximum-likelihood decoding of linear block codes using a trellis, IEEE Trans. Inform. Theory 24 (1978) 76–80.