On the Many Faces of Block Codes

Kaustubh Deshmukh[1], Priti Shankar[2], Amitava Dasgupta[2*], and B. Sundar Rajan[3]

[1] Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India
[2] Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India
[3] Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India

Abstract. Block codes are first viewed as finite state automata represented as trellises. A technique termed subtrellis overlaying is introduced with the object of reducing decoder complexity. Necessary and sufficient conditions for subtrellis overlaying are next derived from the representation of the block code as a group, partitioned into a subgroup and its cosets. Finally a view of the code as a graph permits a combination of two shortest path algorithms to facilitate efficient decoding on an overlayed trellis.

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[9] 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 when compared with hard decision decoding algorithms. That block codes have efficient *combinatorial* descriptions in the form of *trellises* was discovered in

* Presently at Nokia Research Center, Helsinki, Finland

H. Reichel and S. Tison (Eds.): STACS 2000, LNCS 1770, pp. 53–64, 2000.

1974 [1]. Two other early papers in this subject were [19] and [11]. A landmark paper by Forney [3] in 1988 began an active period of research on the trellis structure of block codes. It was realized that the well known Viterbi Algorithm [16] (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 [13]. Trellises have been studied from the viewpoint of linear dynamical systems and also within an algebraic framework [18] [4] [7] [8]. An excellent treatment of the trellis structure of codes is available in [15].

This paper introduces a technique called 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. We finally present a two-stage decoding algorithm where the first stage is a Viterbi algorithm performed on the overlayed trellis. The second stage is an adaption of the A^* algorithm well known in the area of artificial intelligence. It is shown that sometimes decoding can be accomplished by executing only the first phase on the overlayed trellis(which is much smaller than the conventional trellis). Thus overlaying may offer significant practical benefits. Section 2 presents some background on block codes and trellises; section 3 derives the conditions for overlaying. Section 4 describes the new decoding algorithm; finally section 5 concludes the paper.

2 Background

We give a very brief background on subclasses of block codes called linear codes. Readers are referred to the classic text [10].

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

Definition 1. *A* linear block code C *of length* n *over a field* 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. Figure 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 [7] the definition of a trellis for a block code.

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

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

Definition 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 [13] and constructable from a generator matrix for the code [7]. 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 [7]. Figure 2 shows a trellis for the linear code in Figure 1.

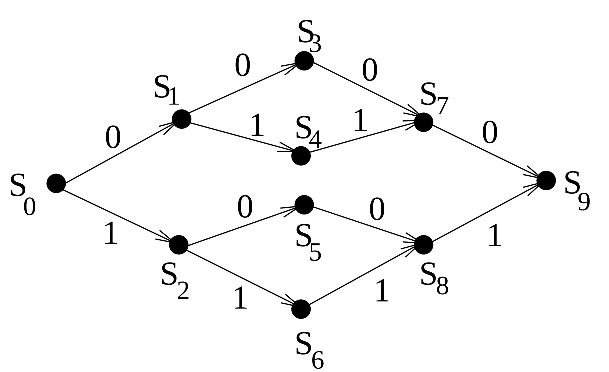

Fig. 2. A trellis for the linear block code of figure 1 with $S_0 = s$ and $S_9 = f$

Willems [18] has given conditions under which an arbitrary block code (which he refers to as a dynamical system) has a unique minimal realization.

Biproper trellises minimize a wide variety of structural complexity measures. McEliece [12] has defined a measure of Viterbi decoding complexity in terms of the number of edges and vertices of a trellis, and has shown that the biproper trellis is 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 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 $i, 0 \leq i \leq n$ 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-complexity profile* of the code is defined as the sequence $(|S_0|, |S_1|, \cdots |S_n|)$. Minimization of S_{max} is often desirable and S_{max} is referred to as the *maximum state-complexity*. 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 1. Let C be the linear $(4, 2)$ code defined by the generator matrix in Figure 1. C consists of the set of codewords $\{0000, 0110, 1001, 1111\}$ and is described by the minimal trellis in Figure 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 figures 3(a) and 3(b) respectively.

The next step is the "overlaying" of the subtrellises as follows. There are as many states at time index 0 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 Figure 4.

Note that overlaying may increase the state-complexity 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 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 figure 3. Overlaying forces subtrellises for

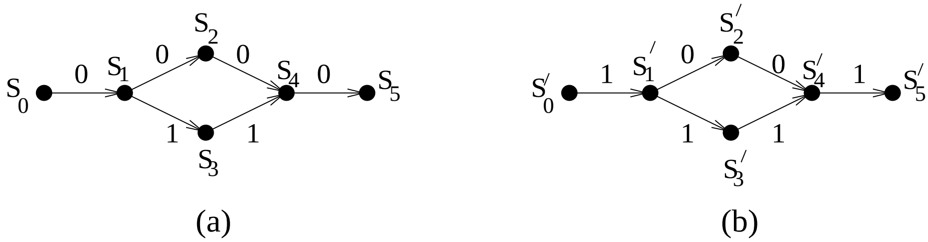

Fig. 3. Minimal trellises for (a) $C_1 = \{0000, 0110\}$ and (b) $C_2 = \{1001, 1111\}$

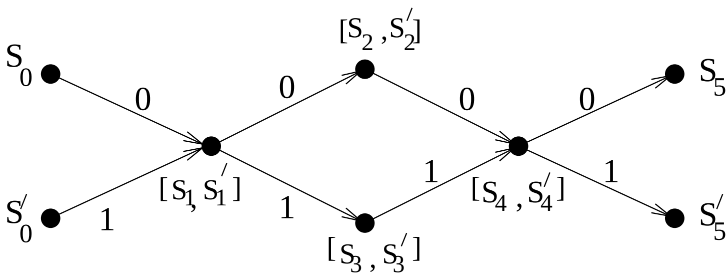

Fig. 4. Trellis obtained by overlaying trellis in figures 3(a) and 3(b)

subcodes to "share" states. Note that the shared trellis is also two way proper, with $S_{max} = 2$ and state-complexity 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 figures 5(a) and 5(b) respectively.

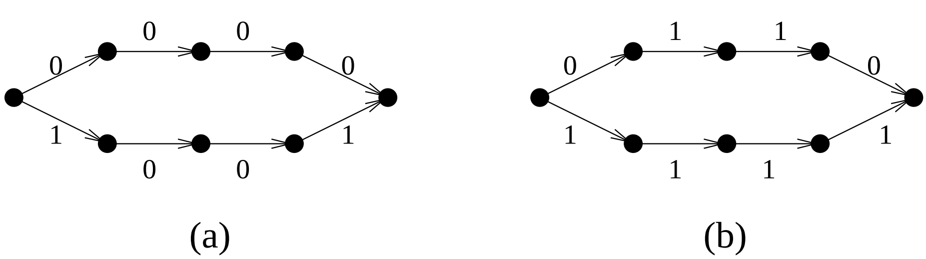

Fig. 5. Minimal subtrellis for (a) $C_1 = \{0000, 1001\}$ and (b) $C_2 = \{0110, 1111\}$

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 for 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 [2]. 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 [17] 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 complexity 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 tailbiting trellises for specific examples from generator matrices in specific forms in [2].

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 $C_1, \ldots C_l$ are cosets of C_0 in C. Let $C_i = C_0 + h_i$ where $h_i, 1 \leq h_i \leq l$ are coset leaders, with C_i having minimal trellis T_i. The subcode 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 subcodes $C_0, C_1, \ldots C_l$ 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 $g_1 g_2 \dots g_n$ in C_0 with $g_1 + h_{i_1}, g_2 + h_{i_2} \dots g_n + h_{i_n}$ in the trellis corresponding to $C_0 + h_i$ where $h_i = h_{i_1} h_{i_2} \dots h_{i_n}$.) We therefore refer to $T_1, T_2, \dots T_l$ as *copies* of T_0.

Definition 3. *An* overlayed *proper trellis is said to exist for C with respect to the partition $C_0, C_1, \dots C_l$ where $C_i, 0 \leq i \leq l$ are subcodes as defined above, corresponding to minimal trellises $T_0, T_1, \dots T_l$ respectively,with $S_{max}(T_0) = N$, iff it is possible to construct a proper trellis T_v satisfying the following properties:*

1. *The trellis T_v has $l+1$ start states labeled $[s_0, \emptyset, \emptyset, \dots \emptyset], [\emptyset, s_1, \emptyset \dots \emptyset] \dots [\emptyset, \emptyset, \dots \emptyset, s_l]$ where s_i is the start state for subtrellis $T_i, 0 \leq i \leq l$.*
2. *The trellis T_v has $l+1$ final states labeled $[f_0, \emptyset, \emptyset, \dots \emptyset], [\emptyset, f_1, \emptyset, \dots \emptyset], \dots [\emptyset, \emptyset, \dots \emptyset, f_l]$, where f_i is the final state for subtrellis T_i, $0 \leq i \leq l$.*
3. *Each state of T_v has a label of the form $[p_0, p_1, \dots p_l]$ where p_i is either $\emptyset$ or a state of $T_i, 0 \leq i \leq l$. Each state of T_i appears in exactly one state of T_v.*
4. *There is a transition on symbol a from state labeled $[p_0, p_1, \dots p_l]$ to $[q_0, q_1, \dots q_l]$ in T_v if and only if there is a transition from p_i to q_i in T_i, provided neither p_i nor q_i is $\emptyset$, for at least one value of i in the set $\{0, 1, 2, \dots l\}$.*
5. *The maximum width of the trellis T_v at an arbitrary time index $i, 1 \leq i \leq n-1$ is at most N.*
6. *The set of paths from $[\emptyset, \emptyset, \dots s_j, \dots \emptyset]$ to $[\emptyset, \emptyset, \dots, f_j, \dots \emptyset]$ is exactly $C_j, 0 \leq j \leq l$.*

Let the *state projection* of state $[p_0, p_1, \dots, p_i, \dots, p_l]$ 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 $C_i, 1 \leq i \leq l$ 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 complexity profile $(m_o, m_1, \dots m_n)$, 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 complexity and t is the last. Note that it is not necessary that this complexity be retained between r and t, i.e., the state complexity 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. The proofs of these are available in [14]

Lemma 1. *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.*

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

Lemma 3. *If trellises T_a and T_b share a state at time index i, then they share all states at time index i.*

Lemma 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.*

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 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.*

Now each of the M/N copies of T has m_i states at level i. Since the width of the overlayed trellis cannot exceed N for $1 \leq i \leq n-1$, 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 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.

Theorem 1. *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 $1 \leq i \leq n-1$ there exist at most N/m_i collections of coset leaders such that
(i) If $1 \leq i < t$, 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.

Corollary 1. *If $M = N^2$ and the conditions of the theorem are satisfied, we obtain a trellis which satisfies the square root lower bound.*

Theorem 1 and corollary 1 answer both the questions about overlayed trellises posed earlier. However, the problem of the existence of an efficient algorithm for the decomposition of the code into a subgroup and its cosets remains open. In the next section we describe the decoding algorithm on an overlayed trellis.

4 Decoding

Decoding refers to the process of forming an estimate of the transmitted codeword $\mathbf{x}$ from a possibly garbled received version $\mathbf{y}$. The received vector $\mathbf{y}$ consists of a sequence of n real numbers, where n is the length of the code. 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 [16] 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, $0 \leq i \leq l$.

4.2 The A^* Algorithm

The A^* algorithm is well known in the literature on artificial intelligence [6] and is a modification of the Dijkstra shortest path algorithm . That the A^* algorithm can be used for decoding was demonstrated in [5]. 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 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 $h(u, f) \leq L_T(u, f)$, and such that $h(u, f)$ satisfies the following inequality, i.e, for u a predecessor of v, $l(u, v) + h(v, f) \geq h(u, f)$. 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 $l(u, v) \geq 0$ for edge (u, v) in E, a source vertex s and a destination vertex f.
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
 $S \leftarrow \emptyset, \quad \bar{S} \leftarrow \{s\}, \quad p(s) \leftarrow 0, \quad P(s) \leftarrow ()$
 repeat
  Let u be the vertex in $\bar{S}$ with minimum value of $p(u) + h(u, f)$.
  $S \leftarrow S \cup \{u\}; \quad \bar{S} \leftarrow \bar{S} \setminus \{u\};$
  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

Decoding on an overlayed trellis needs at most two phases. In the first phase, a conventional Viterbi algorithm is run on the overlayed trellis T_v . The aim of this phase is to obtain estimates $h()$ for each node, which will subsequently be 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. The two phases are described below. (All assertions in italics have simple proofs given in [14]).
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*, $1 \leq j \leq N$. *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.* Initialize a sequence P_j for each residual trellis T_j to the empty sequence. P_j, in fact stores the current candidate for the shortest path in trellis T_j. Let the estimate $h(s_j, f_j)$ associated with the empty path 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 $h()$ 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 $h(s_i, f_j) - cost(survivor(u))$ where $cost(survivor(u))$ is the cost of the survivor obtained in the first phase. $h()$ *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.

5 Conclusions

This paper offers a new perspective from which block codes may be fruitfully viewed. A technique called subtrellis overlaying is proposed , which reduces the size of the trellis representing the block code. Necessary and sufficient conditions for overlaying are derived from the representation of the code as a group. Finally a decoding algorithm is proposed which requires at most two passes on the overlayed trellis. For transmission channels with high signal to noise ratio, it is likely that decoding will be efficient. This is borne out by simulations on a code called the hexacode[2] on an additive white Gaussian noise(AWGN) channel, where it was seen that the decoding on the overlayed trellis was faster than that on the conventional trellis for signal to noise ratios of 2.5 dB or more[14]. Future work will concentrate on investigating the existence of an efficient algorithm for finding a good decomposition of a code into a subgroup and its cosets, and on obtaining overlayed trellises for long codes.

References

1. L.R.Bahl, J.Cocke, F.Jelinek, and J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inform. Theory* **20**(2), March 1974, pp 284-287.
2. A.R.Calderbank, G.David Forney,Jr., and Alexander Vardy, Minimal Tail-Biting Trellises: The Golay Code and More, *IEEE Trans. Inform. Theory* **45**(5) July 1999,pp 1435-1455.
3. G.D. Forney, Jr.,Coset codes II: Binary lattices and related codes, *IEEE Trans. Inform. Theory* **36**(5), Sept. 1988,pp 1152-1187.
4. G.D. Forney, Jr. and M.D. Trott, The dynamics of group codes:State spaces, trellis diagrams and canonical encoders, *IEEE Trans. Inform. Theory* **39**(5) Sept 1993,pp 1491-1513.
5. Y.S.Han, C.R.P.Hartmann, and C.C.Chen, Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes, *IEEE Trans. Inform.Theory* **39**(5),Sept. 1993,pp 714-729.
6. P.E. Hart, N.J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Solid-State Circuits* **SSC-4**, 1968, pp 100-107.
7. F.R.Kschischang and V.Sorokine, On the trellis structure of block codes, *IEEE Trans. Inform Theory* **41**(6), Nov 1995,pp 1924-1937.
8. F.R.Kschischang, The trellis structure of maximal fixed cost codes, *IEEE Trans. Inform Theory* **42**(6), Nov. 1996, pp 1828-1837.
9. D.Lind and M.Marcus, *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, 1995.
10. F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1981.
11. J.L.Massey, Foundations and methods of channel encoding, in *Proc. Int. Conf. on Information Theory and Systems* **65**(Berlin, Germany) Sept 1978.
12. R.J. McEliece, On the BCJR trellis for linear block codes, *IEEE Trans. Inform Theory* **42**, 1996, pp 1072-1092
13. D.J. Muder, Minimal trellises for block codes, *IEEE Trans. Inform Theory* **34**(5), Sept 1988,pp 1049-1053.
14. Amitava Dasgupta, Priti Shankar, Kaustubh Deshmukh, and B.S.Rajan, *On Viewing Block Codes as Finite Automata*, Technical Report IISc-CSA-1999-7, Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India, November, 1999.
15. A.Vardy, Trellis structure of codes, in *Handbook of Coding Theory*, R.A.Brualdi, W.C. Huffman, V.S. Pless, Eds., Vol.2, Chap. 24, Elsevier, 1998.
16. A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Trans. Inform Theory* **13**, April 1967, pp 260-269.
17. N.Wiberg, H.-A. Loeliger and R.Kotter, Codes and iterative decoding on general graphs, *Euro. Trans. Telecommun.,***6** pp 513-526, Sept 1995.
18. J.C. Willems, Models for Dynamics, in *Dynamics Reported*, **2**, U. Kirchgraber and H.O. Walther, Eds. New York: Wiley, 1989, pp 171-269.
19. J.K. Wolf, Efficient maximum-likelihood decoding of linear block codes using a trellis, *IEEE Trans. Inform Theory* **24**(1), January 1978, pp 76-80.