Soft Trellis-Based Decoder for Linear Block Codes

Yuval Berger and Yair Be'ery, *Member, IEEE*

***Abstract*—A systematic design of a trellis-based maximum-likelihood soft-decision decoder for linear block codes is presented. The essence of the decoder is to apply an efficient search algorithm for the error pattern on a reduced trellis representation of a certain coset. Rather than other efficient decoding algorithms, the proposed decoder is systematically designed for long codes, as well as for short codes. Computational gain of up to 6 is achieved for long high-rate codes over the well-known trellis decoder of Wolf. Efficient decoders are also obtained for short and moderate length codes.**

***Index Terms*—Trellis, maximum-likelihood decoding, soft-decision decoding, block codes.**

I. Introduction

SEVERAL efficient soft-decision maximum-likelihood (ML) decoding algorithms for block codes have been proposed over the years [1]–[7]. The majority of these algorithms [2]–[7] may be designed in practice only for short and moderate length codes; coset decoding schemes [2], [4]–[6] oblige the selection of a specific subcode. The related decoding complexity is considerably sensitive to this selection. However, no general procedure for selecting a "good" subcode is available. In the construction of a reduced ordered list decoder [7], error patterns are represented by sets of indices which correspond to the least confidence values. This process becomes quite cumbersome for long codes. In contrast to all the methods above, the systematic trellis-based decoder proposed by Wolf [1] may be straightforwardly designed for any code since it requires only the knowledge of its parity check matrix.

In this paper we introduce a systematic design of an optimal ML trellis-based decoder. Structural and algebraic properties of codes are incorporated in this design, thus yielding increased efficiency. However, the decoder is generally designed for wide classes of codes. The decoding complexity is measured as usual [1]–[7] by the number of real addition-equivalent operations. Computational gain of up to 6 is achieved in the proposed decoder for long high-rate codes over Wolf's decoder [1] (which is the only alternative currently known for these codes, to the best of our knowledge). Relatively low decoding complexity is also obtained for many short and moderate length codes. In view of this design we also show that a suboptimal bounded-distance decoder may be straightforwardly constructed.

Manuscript received June 3, 1992; revised April 2, 1993.

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

IEEE Log Number 9401370.

ML soft-decision decoding of an (n, k) binary block code C can be stated by the rule [3]

$$\max_{c \in C} \sum_{i=1}^{n} (-1)^{c_i} \mu_i, \tag{1}$$

where $\mu_i \triangleq \log(p(\theta_i/0)/p(\theta_i/1))$, θ_i is the received analog channel information, and $p(\theta_i/0)$, $p(\theta_i/1)$ are the channel transition probabilities. An equivalent formula based on the error pattern, which is a generalization of the Wagner rule [5], is

$$\min_{e \in K_r} \sum_{i=1}^{n} e_i |\mu_i|, \tag{2}$$

where K_r is the coset including the binary n-tuple obtained from θ by using a hard decision.

The proposed decoder operation is based on the following steps:

- A certain coset, which represents the coset of the received word, is determined.
- The error pattern is found by an efficient ML search over a punctured trellis representation of the coset.
- The received word is corrected with the aid of the error pattern.

We shall briefly state the basics of trellis theory related to block codes, and then proceed with an outline of the paper. Let C denote a code over $GF(q)$. The trellis diagram for C consists of $n + 1$ levels denoted by V_i, $i = 0, 1, \cdots, n$. $|V_i|$ is the number of states at level V_i. Each branch of the trellis is labeled by $0, 1, \cdots, q - 1$, and each path corresponds to a codeword. Following the notation in [8], a trellis is called *proper* if V_0 consists of a single state v_0, every state lies on some length-n path, and no two length-i paths that originate in v_0 designate the same i-tuple. Hereafter we refer only to proper trellises. Denote by H the parity check matrix of C, and let the $(n - k)$-tuple h_i be the ith column of H, $i = 1, 2, \cdots, n$. Consider the following trellis construction [1], which is exhibited by an arbitrary state $v \in V_i$ with the following label from $GF(q)^{n-k}$,

$$v = \sum_{j=1}^{i} b_j h_j, \tag{3}$$

where the path labeled by $(b_1, b_2, \cdots, b_n)$ corresponds to a codeword. Obviously $|V_0| = 1$ and $|V_n| = 1$. Let v_0, v_n be

the initial and final states which belong to V_0, V_n, respectively. We assume that $v_0 = (0,0,\cdots,0)^t$, and consequently $v_n = (0,0,\cdots,0)^t$. The trellis obtained by this construction is a unique proper *minimal trellis* [4], [8], i.e., $|V_i|$ is minimal for $i = 0,1,\cdots,n$.

For a minimal trellis, define the state space dimension $s_i \triangleq \log_q |V_i|$. Define the *minimal trellis size* index $s \triangleq \max\{s_i\}$. The minimal trellis size is usually used [1], [4], [8], [9] as a measure of the decoding complexity of the code with respect to trellis-based decoding. A general upper bound on s [1] is $s \leq \min(k, n-k)$. It is well known that the minimal trellis size depends on the order of the code coordinates [4]. The s index, related to the optimal permutation (in the sense of minimal trellis size), is called the *absolute minimal trellis size* [9], and denoted by $s(C)$. A further formalization of trellis theory has been recently presented by Forney and Trott [12].

The major stages of the proposed decoder construction, which are performed only once, are presented in Sections II, III, and V. The decoding algorithm is given in Section IV. In order to simplify the presentation, Sections III–V are presented for binary codes. The essence of these sections is summarized next.

In Section II we present the construction of a minimal trellis representation for any coset of code C over $GF(q)$. Based on similarities between these trellises and the code trellis, we exploit known results for s and $s(C)$ [1], [4], [8]–[11], originally related to the code trellis.

In Section III we show that coset trellises generally possess many redundant branches and states with respect to ML decoding. An efficient puncturing algorithm is devised. The revised trellis is called a *punctured trellis*.

The decoding algorithm (which is the process of correcting the received channel word) is presented in Section IV. It is essentially a searching scheme over a punctured trellis. This search algorithm is a modification of the well-known Viterbi algorithm [13], where additional redundancy in the coset trellis is utilized. The computational complexity is evaluated for each of the punctured minimal coset trellises, as part of the decoder construction. It is exploited in the next construction stage.

Berlekamp [14] described a method of projecting the complete set of cosets of the extended (128, 106, 8) BCH code onto a small set. A generalization of this concept is presented in Section V. According to the complexity calculated before and by the use of coset projections, a small set of *coset representatives* is determined. Thus, the overall decoder's computational complexity is decreased. Moreover, a substantial reduction of the storage complexity is achieved. This concludes the decoder construction.

The proposed decoder has been designed for various codes. Several examples are reported in Section VI. For many cases, the complexity is lower than that of the currently known decoding algorithms. For long high-rate codes, a gain of approximately 6 is exhibited over Wolf's method [1]. Also, the implications of the proposed decoding scheme for practical bounded-distance decoding are discussed.

II. Minimal Coset Trellis

Definition 1: For an (n,k) code over $GF(q)$ with parity check matrix H, define a trellis T_r, $r \in GF(q)^{n-k}$, with initial state $v_0 = (0,0,\cdots,0)^t$ by labeling the states as in (3) such that the final state is $v_n = r$.

Clearly T_0 is the minimal code trellis of C ($\mathbf{0}$ is the all-zero vector). Also $a = (a_1, a_2, \cdots, a_n)$ is a word from the coset K_r, with syndrome r, if and only if it corresponds to a path in T_r. Consider a mapping $\tilde{v} = v + \sum_{j=1}^{i} a_j h_j$ from a state $v \in V_i$ in T_0 to a state $\tilde{v} \in \tilde{V}_i$ in T_r. Notice that $\tilde{v} = \sum_{j=1}^{i}(b_j + a_j)h_j \triangleq \sum_{j=1}^{i} u_j h_j$, where $u \in K_r$, i.e., $\sum_{j=1}^{n} u_j h_j = r$ and $\tilde{v}$ indeed belongs to $\tilde{V}_i$. Let $b^{(i)}$ denote the truncated sequence $(b_1, b_2, \cdots, b_i)$. A related mapping is then established from the path $b^{(i)}$ to $u^{(i)}$. A similar mapping from any state $\tilde{v} \in \tilde{V}_i$ in T_r to a state $v \in V_i$ in T_0 may be defined by $v = \tilde{v} - \sum_{j=1}^{i} a_j h_j$. In fact, this is the inverse of the previous mapping. We conclude that there is a one-to-one correspondence between the states and branches of T_0 and T_r. The next theorem readily follows.

Theorem 1: T_r is a minimal trellis representation for K_r, and all the minimal trellis representations of K_r are isomorphic, $r \in GF(q)^{n-k}$.

T_r will be called the *minimal coset trellis* of K_r. There is no general formula for the absolute minimal trellis size of a block code, nor is there a solution for the problem of finding the optimal permutation for which $s = s(C)$. Nevertheless, a few studies [1], [4], [8]–[12] contributed to a better understanding of the trellis properties, and also provided solutions for the above problems in some cases. We shall outline two results. The first is a general upper bound on $s(C)$ [9],

$$s(C) \leq \min(k - J_\nu + \nu + 1, n - k - J_{\nu^\perp} + \nu^\perp + 1), \quad (4)$$

where J_ν is the dimension of a subcode of C with *contraction index* ν (ν is the difference between the number of pairwise linearly independent columns of the subcode and its dimension J_ν [5]), and $J_{\nu^\perp}$ is a similar parameter related to the dual code of C. Furthermore, the proper permutation for achieving the bound is provided in [9]. The contraction concept was introduced in [5] as part of a *coset decoding* method for block codes. The problem of estimating the maximal J_ν was further analyzed in [16] where several bounds were derived. For $\nu = 0$, J_0 becomes the number of *zero-concurring* codewords of C (in which at most one word has a nonzero element in any code coordinate [3]) and can be simply evaluated [15].

Additional constructive bounds for various classes of codes are also provided in [9]. The main class is the Reed–Muller codes RM (r, m), of order r and length 2^m, for which

$$s(\mathrm{RM}(r,m)) = \sum_{i=0}^{\tilde{r}} \binom{m-2i-1}{\tilde{r}-i}, \quad (5)$$

where $\tilde{r} \triangleq \min(r, m-r-1)$. This formula was originally presented in [9] as the exact minimal trellis size index

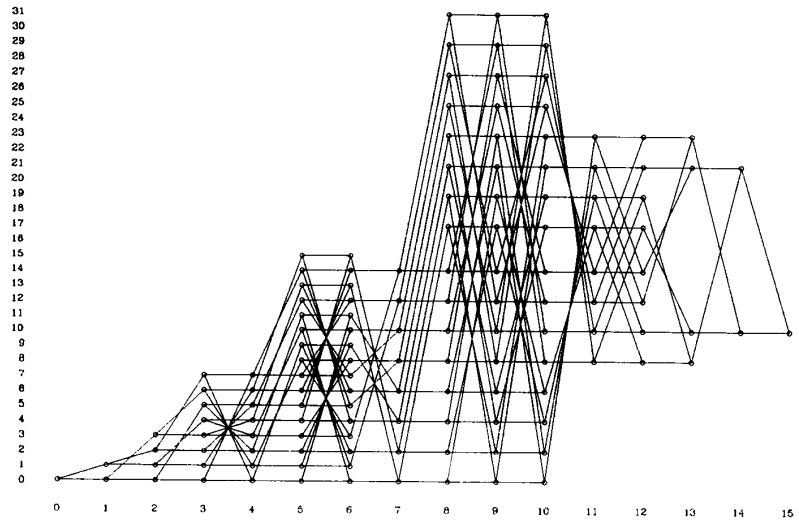

Fig. 1. T_r for the BCH (15, 10, 4) code, $r = (01010)^t$.

related to the *squaring construction* (SC) design of these codes [4], and an upper bound on $s(\mathrm{RM}(r, m))$. Due to the fact that this permutation is optimal [11] in the sense of the minimal trellis size at every level, the identity in (5) is implied.

The first stage of the decoder construction is to determine the appropriate permutation, with the aid of the results above, and construct the minimal coset trellises. Notice that results such as (4) and (5) were originally obtained for the code trellis. However, according to Theorem 1, the results apply for the coset trellises as well.

Example 1: Let C be the expurgated Hamming (15, 10, 4) code. We shall determine the permutation by observing the SC design of $C_1 = \mathrm{RM}(2, 4)$, with parity check matrix

$$H_1 = \begin{bmatrix} 1111111111111111 \\ 0101010101010101 \\ 0011001100110011 \\ 0000111100001111 \\ 0000000011111111 \end{bmatrix}.$$

From (5), the minimal trellis size of C_1 is $s = 4$. C may be constructed from C_1 by puncturing any column from H_1. For example, by puncturing the eighth column we obtain

$$H = \begin{bmatrix} 111111111111111 \\ 010101001010101 \\ 001100100110011 \\ 000011100001111 \\ 000000011111111 \end{bmatrix}.$$

We have systematically found a permutation for the (15, 10) code for which $s = 4$ [9]. The minimal coset trellises may be now constructed according to Definition 1, based on this permutation. For example, the minimal coset trellis of K_r, $r = (01010)^t$, is illustrated in Fig. 1. Each row is numbered according to the radix-2 representation of its states. Horizontal and diagonal branches are labeled by 0 and 1, respectively.

The use of a proper permutation in the Wolf decoder design [1] leads to an immediate gain in the computational decoding complexity. It can be easily verified that the (worst case) decoding complexity of the original Wolf method is either $2^{n-k}(6k - 3n + 5) - 5$ or $2^k(3n - 6k + 5) - 5$ for the cases of $n \le 2k$ and $n > 2k$, respectively. It can be similarly shown that the Viterbi decoding complexity with respect to a specific permutation with minimal trellis size index s is at most

$$2^s(3n - 6s + 5) - 5. \tag{6}$$

Consequently, the complexity gain for high-rate codes over the Wolf decoder is at least

$$\frac{2^{n-k}(6k - 3n + 5) - 5}{2^s(3n - 6s + 5) - 5}. \tag{7}$$

For instance, in Example 1 this gain is $(2^5 \cdot 20 - 5)/(2^4 \cdot 26 - 5) = 635/411 \simeq 1.55$.

III. Punctured Minimal Coset Trellis

In this section the punctured minimal coset trellises are established. We describe the redundancy possessed by the coset trellises, and provide efficient puncturing methods.

Consider the set I_e of columns from H that correspond to the nonzero elements of a pattern e. Formula (2) reveals the well-known property [5] that error patterns whose corresponding columns in H form a linearly dependent set may be eliminated from the soft ML decoding process. This fact will be exploited for puncturing the coset trellises. Branches that lie only on paths related to such dependent patterns are called *dependency branches* (DB). *Dependency states* (DS) are similarly defined. The general determination of the DB or DS requires an exhaustive search over the trellis, which is impractical for long codes. Therefore several specific types of DB are

defined in the following, for which much simpler puncturing algorithms are provided. Each type is proved to comply with the DB definition.

A. DB—Type I

A particular set of patterns that correspond to dependent column sets are words with Hamming weight greater than $n - k$ [5]. A branch (state) in a coset trellis is called a DB—Type I (DS—Type I) if the Hamming weight of every path in the trellis which includes this branch (state) is greater than $n - k$.

Lemma 1: For any T_r, the following statements are equivalent:

a) v in T_r is a DS—Type I.
b) All the branches that enter v are DB—Type I.
c) All the branches that leave v are DB—Type I.

Proof: Suppose (a) holds. Any DB—Type I which either enters or leaves v lies on a path that obviously passes through v, and therefore is of weight greater than $n - k$. Thus (a) $\Rightarrow$ (b), (c). Similarly, every path that passes through v must include one of the branches entering v, and one of those leaving v. Thus, (b) $\Rightarrow$ (a), (c) $\Rightarrow$ (a). □

DB—Type I may be purged from the trellis during the decoder construction process, ensuring also the removal of all DS—Type I. Let $W_p(v)$ denote the *past weight* of v, namely, the minimum among the Hamming weights of all paths from v_0 to v. Similarly, $W_f(v)$ denotes the *future weight* of v, referring to paths from v to v_n. A branch ϕ that connects states $v \in V_i$ and $z \in V_{i+1}$, $i = 0, 1, \cdots, n - 1$, is a DB—Type I if and only if the sum $W_p(v) + W_f(z) + W(\phi)$ is greater than $n - k$, where $W(\phi)$ denotes the weight of ϕ. The following algorithm (of computational complexity $O(n2^s)$ detects and purges all the DB—Type I.

Algorithm A

1. $i \leftarrow 0$, $W_p(v_0) \leftarrow 0$, $W_f(v_n) \leftarrow 0$.
2. $i \leftarrow i + 1$.
3. For each state $v \in V_i$, $W_p(v) \leftarrow \min(W_p(z_0), W_p(z_1) + 1)$, where $z_0, z_1 \in V_{i-1}$ are the states connected to v by branches with labels 0, 1, respectively.
4. If $i < n$ then go to step 2.
5. Repeat steps 2–4 for $W_f(v)$, but this time start with $v(n)$ and step backwards.
6. For each branch ϕ in the trellis connecting $v \in V_i$ and $z \in V_{i+1}$, $i = 0, 1, \cdots, n - 1$, if $W(\phi) + W_p(v) + W_f(z) > n - k$ then purge ϕ.
7. End.

Theorem 2: A branch is purged from the trellis by Algorithm A if and only if it is a DB—Type I. The resulting punctured trellis is proper.

Proof: Clearly, the condition in step 6 of the algorithm for purging a branch is fulfilled only by a DB—Type I. We shall prove now that the punctured trellis is proper. Since we started with a proper trellis, this property may be violated now only if some states do not lie now on length-n paths. However, Lemma 1 assures that if the incoming branches of a state are purged, then also the outgoing branches are detached, and *vice versa*. Thus states either remain on length-n paths or completely detached from the trellis.

The other part of the theorem states that every DB—Type I in the trellis is purged by this algorithm. This may be false only if a branch which was originally a non-DB—Type I becomes such (by definition) in the new trellis resulting from Algorithm A. This may occur only if paths of weight not greater than $n - k$ have been eliminated from the trellis. However, this contradicts the fact that only DB have been deleted. □

Consider now a code $C(n, k, d)$ such that $d \geq n - k$. Clearly any possible dependency branch lies on paths of Hamming weight greater than d, and thus it is DB—Type I. This is formally stated in the next lemma.

Lemma 2: In a trellis of coset K_r of code $C(n, k, d)$ where $n - k + 1 \geq d \geq n - k$, any dependency branch is a DB—Type I.

B. DB—Type II

Let DB—Type II in T_r be the 1-labeled branches entering states 0 or leaving states r.

Lemma 3: DB—Type II are dependency branches in T_r, $r \in GF(2)^{n-k}$.

Proof: Any path in T_r from $v_0 = \mathbf{0}$ to a state with label 0 must have nonzero-labeled branches which correspond to columns with zero sum, i.e., these columns are dependent. Any 1-labeled branch which enters a zero-labeled state belongs only to such paths, and thus is a dependency branch. A similar proof applies also for 1-labeled branches that start at a state with label r. □

The number of DB—Type II is $O(n)$. These branches are easily found in the trellis, yet their elimination is effective only for short or moderate length codes.

C. DB—Type III

A path in T_r from a state $v_1 \in V_i$ to a state $v_2 \in V_{i+l}$, $i = 0, 1, \cdots, (n - l)$, may possibly possess the following property: Any state lying on the path, except for v_2, has only a single outgoing branch (which lies on this path). Denote by ϕ_1 the first branch on this path, leaving v_1. Suppose that the sum of columns from H corresponding to the nonzero branches on this path is r. Also this path lies on length-n paths whose sum of corresponding H columns is r. Hence, each such length-n contains an internal length-l path, $1 \leq l < n$, from v_1 to v_2, with the same property. If $v_1 \neq \mathbf{0}$ (or equivalently $v_2 \neq r$), then these length-n paths are related to dependent sets of H columns, and ϕ_1 is a DB. A similar case involves also a path between states v_1 and v_2, where each state in the path contains a single incoming branch. The incoming branch of v_2 is denoted by ϕ_2. If this path is related to columns of sum r, and $v_1 \neq \mathbf{0}$ (or $v_2 \neq r$), then similarly ϕ_2 is a DB. ϕ_1 and ϕ_2 are called DB—Type III. Examples of both cases of DB—Type III and the relevant length-l paths appear in Fig. 2. (In Fig. 2(a) the syndrome

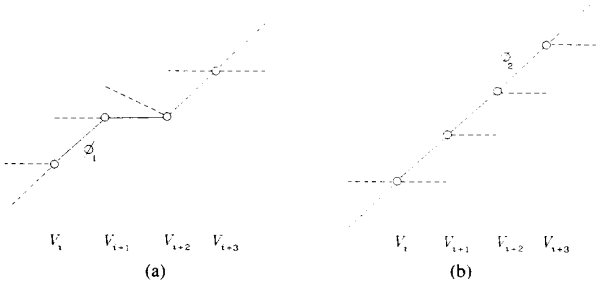

Fig. 2. Examples of DB—Type III.

equals $h_{i+1} + h_{i+3}$, while in Fig. 2(b) the syndrome equals $h_{i+1} + h_{i+2} + h_{i+3}$.)

Hence, DB—Type III are dependency branches and may be purged from T_r, $r \in GF(2)^{n-k}$.

Since d is the minimal cardinality of a dependent column set, the length-l paths above may contain no more than $n - k - d$ 1-labeled branches, provided that DB—Type I have been deleted. Thus the complexity of the search for DB—Type III is generally $O(2^s n(n - k - d))$. As before, we assume that states and branches that no longer belong to length-n paths after purging the dependency branches are discarded as well.

D. Generalized DB—Type I

Additional redundancy in the coset trellises, besides dependency branches, is embedded within the set of error patterns of weight $n - k$. This fact was recently pointed out by Snyders [7] in the context of a reduced ordered list decoder. We present a simple and direct proof to this property in the next theorem.

Theorem 3: At most one error pattern of weight $n - k$ has to be considered in a ML soft-decision decoding process.

Proof: Consider the basis to the linear space $GF(2)^{n-k}$ formed by the set B of independent columns from H with minimal sum of the corresponding confidence values $|\mu_i|$. There is a particular subset $I_{\hat{e}} \subseteq B$ whose linear sum equals the syndrome r, where $\hat{e}$ is an error pattern. Let e be an error pattern with weight $n - k$. If I_e contains a subset of linearly dependent columns then, from (2), e may be eliminated. Otherwise, I_e is also a basis for $GF(2)^{n-k}$. From the definition of B, the sum of confidence values of $I_{\hat{e}}$ is not greater than that of I_e. Consequently, e may be eliminated, unless of course $e = \hat{e}$. □

Definition 2: A branch (state) in a coset trellis is called a *generalized DB—Type I* (*DS—Type I*) if every path in the trellis which includes this branch (state) has a Hamming weight greater than $n - k - 1$.

In order to utilize Theorem 3 in the decoder, we may purge from the trellis all the generalized DB—Type I. This is performed by Algorithm A, where the expression $n - k$ in step (6) is replaced by $n - k - 1$. It can be easily verified that Lemma 1 and Theorem 2 apply also for generalized DB—Type I. Still, we must consider also the error pattern $\hat{e}$. If the weight of $\hat{e}$ is less than $n - k$, the word is already included in the trellis. Otherwise, we should evaluate the sum of confidence values involved with $\hat{e}$, and compare to the result of the trellis search. This requires the partial sorting of $\{|\mu_i|\}$ by which the $n - k - 1$ minimal confidence values, related to an independent column set, are found. Using a *binary tree sorting* technique [17, p. 142], the minimal confidence value is found by $n - 1$ comparisons, and the successive minimal values are found by no more than $\lceil \log_2 n \rceil - 1$ comparisons each. Evaluation of the sum of confidence values and the comparison requires $n - k$ operations. The total expense is

$$2n - k - 1 + (n - k - 2)(\lceil \log_2 n \rceil - 1). \quad (8)$$

The process of puncturing the generalized DB—Type I is not necessarily advantageous due to the cost of (8). Both cases may be checked as part of the decoder construction, and the preferable alternative is selected.

Example 2: In this example we demonstrate the minimal and punctured minimal coset trellises of the Hamming $(7, 4)$ code. The code is permuted according to increasing order of the radix-2 representation of the vectors in $GF(2)^3$. Hence

$$H = \begin{bmatrix} 1010101 \\ 0110011 \\ 0001111 \end{bmatrix}.$$

The minimal coset trellis for K_r, $r = (001)^t$, is displayed in Fig. 3(a). A punctured trellis is illustrated in Fig. 3(b), where the DB are deleted. In Fig. 3(c), generalized DB—Type I are also purged.

IV. The Search Algorithm and Decoding Complexity

In a conventional Viterbi algorithm for the code trellis which is governed by (1), the metric $M(v)$ associated with a state $v \in V_i$, with incoming branches c_v, $\bar{c}_v$ from states $z_1, z_2 \in V_{i-1}$, respectively, is

$$M(v) \leftarrow \max\left((-1)^{c_v}\mu_i + M(z_1), (-1)^{\bar{c}_v}\mu_i + M(z_2)\right). \quad (9)$$

The execution of (9) requires three real addition-equivalent operations. If one of the branches is missing in the punctured trellis, one operation at the most suffices. In the coset trellis, according to (2), the corresponding expression for a state v is

$$M(v) \leftarrow \min\left(e_v|\mu_i| + M(z_1), \bar{e}_v|\mu_i| + M(z_2)\right), \quad (10)$$

where $e_v, \bar{e}_v$ are the branches entering v. Obviously either e_v or $\bar{e}_v$ is zero. By this observation, the number of operations involved with each state may be reduced to two (at the most). By this simple modification of the conventional Viterbi scheme, we have managed to reduce approximately one-third of the total number of operations in the searching algorithm. The following example exhibits this fact. In this example we also exploit the fact that

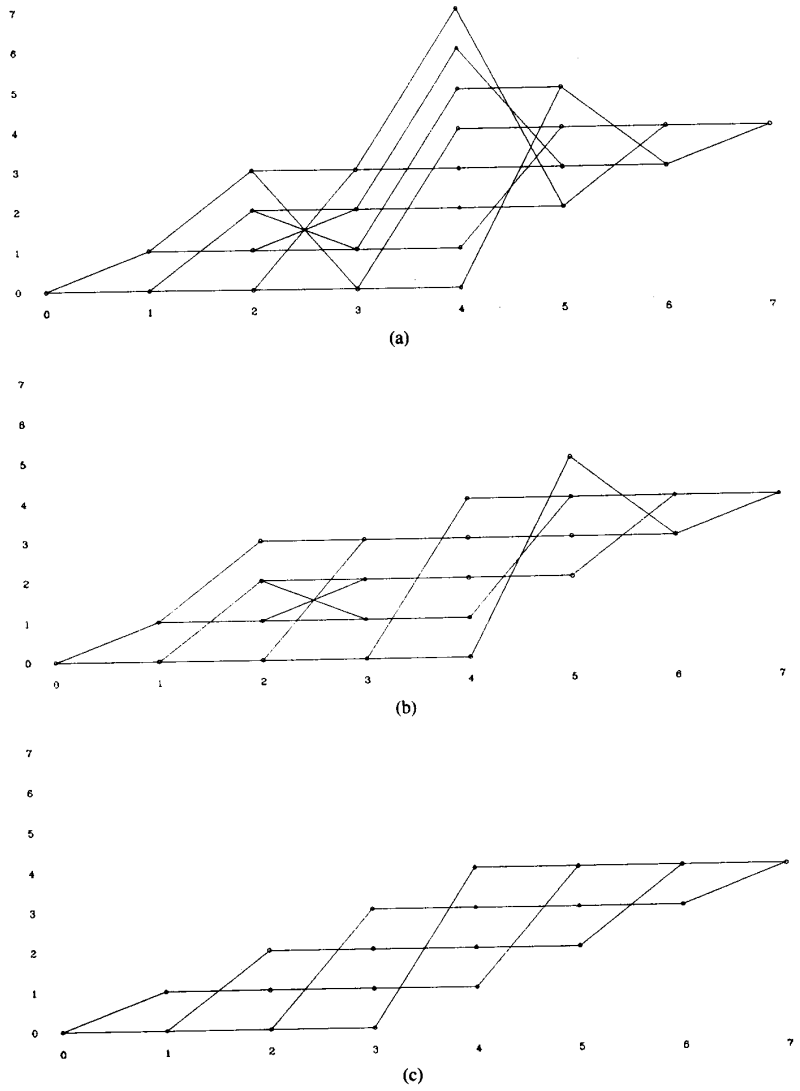

Fig. 3. T_r diagrams for the Hamming (7, 4, 3) code, $r = (001)^t$.

branches with label 1 which leave a state $v = \mathbf{0}$ in any level, called *trivial branches*, do not require the trivial addition $M(v) + |\mu_i| = |\mu_i|$ in (10) (clearly this is significant only for short codes).

Example 3: Consider the punctured coset trellis of the Hamming code (7, 4) described in Fig. 3(b). A total number of 14 add/compare operations is required, where five trivial branches are encountered. Furthermore, two operations may be eliminated by the following procedure: Suppose now that the maximal among $|\mu_1|, |\mu_2|, |\mu_3|$ is found with the expense of two operations. If $|\mu_1|$ is the maximal, then necessarily $M(v_2) \leftarrow |\mu_2|$, $M(v_3) \leftarrow |\mu_3|$, $v_2, v_3 \in V_3$ (v_j is a state in row j). Alternatively, if $|\mu_2|$ is the maximal, then necessarily $M(v_1) \leftarrow |\mu_1|$, $M(v_3) \leftarrow |\mu_3|$, $v_1, v_3 \in V_3$. The last case is when $|\mu_3|$ is the maximal, and then $M(v_1) \leftarrow |\mu_1|$, $M(v_2) \leftarrow |\mu_2|$, $v_1, v_2 \in V_3$. In any case, four operations are eliminated. Thus we decode the Hamming (7, 4) code with a total number of 12 real-equivalent operations, which is a new "record" for soft-decision decoding of this code (previous result is 14 [5]). Note that only a single punctured coset trellis has to be stored for the decoder (more detailed explanation appears in Section V). Thus a low storage complexity is also obtained, as compared to other schemes such as in [1].

Additional modifications of the search scheme are based on specific structures and properties of the coset trellis. Consider the structure in Fig. 4(a) which may appear in the trellis. Generally, the calculation of $M(v_3), M(v_4)$ requires four operations.

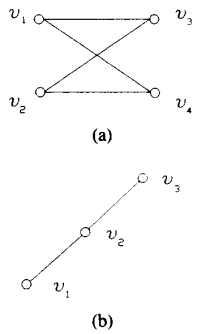

Fig. 4. Trellis structures with redundancy.

Theorem 4: The metrics $M(v_3), M(v_4)$ in Fig. 4(a) may be calculated in three operations in a coset trellis with non-negative confidence values.

Proof: Suppose that $M(v_1) \le M(v_2)$. From (10), necessarily $M(v_3) \leftarrow M(v_1)$. Similarly, if $M(v_2) \le M(v_1)$, evidently $M(v_4) \leftarrow M(v_2)$. Thus instead of the regular computation of the metrics as described before, we may compare $M(v_1), M(v_2)$, and then automatically update either $M(v_3)$ or $M(v_4)$, with the cost of only one operation. Hence three operations are required at the most for evaluating both $M(v_3)$ and $M(v_4)$. □

Consider now the structure of Fig. 4(b). Suppose $v_3 \in V_i$, $i = 2, 3, \cdots, n$. The sum $\rho \triangleq |\mu_{i-1}| + |\mu_i|$ may be computed once in a precomputation stage. Then $M(v_3)$ (or its equivalent for any other such structures) may be calculated with the expense of one operation, i.e., $M(v_3) \leftarrow \rho + M(v_1)$. The benefit of this scheme is clearly proportional to the number of occurrences of this structure in the same trellis levels. For N such occurrences, $N - 1$ real addition-equivalent operations are eliminated as compared to the Viterbi algorithm.

We have described so far in this section an improved searching algorithm over a coset trellis, and the number of operations involved. During the decoder construction, this algorithm is simulated for *all* the punctured coset trellises of the code. The computational complexity is then evaluated on behalf of the last stage of the decoder construction, which will be described in the next section.

The complexity for each coset obviously depends on the chosen permutation and the structure of the punctured trellis. A general formula for the bit level trellis size of neither the original trellis nor the punctured trellis is known. This fact prevents an exact analytic formula of the decoding complexity for the proposed decoder design. However, asymptotic analysis for high-rate codes may be developed. The proposed search algorithm, comprised of (10) and Theorem 4, yields the asymptotic lower bound $(\frac{3}{2})(\frac{4}{3}) = 2$ on the complexity gain over the Viterbi algorithm. It follows from (6) that the proposed decoder complexity is thus asymptotically upper bounded by

$$2^{s-1}(3n - 6s + 5).$$

For example, in the case of extended Hamming $(2^m, 2^m - m - 1, 4)$ codes, (5) implies $s = m$, and the asymptotic gain over Wolf's decoder [1] is thus lower bounded by 4. Numeric calculation (see Table III) shows, however, an asymptotic gain of about 6. This gap evolves from two factors: the trellis size at the bit levels which does not uniformly equal s, and the complexity reduction which results from the puncturing process. Note that the relative gain of each part of this design may vary for different codes.

V. The Set of Coset Representatives

The purpose of the last construction stage is to select a set of *coset representatives* for the code cosets. The main goals are to minimize the cardinality of this set and yet to obtain cosets with relatively low computational decoding complexity as compared to the rest of the cosets. Two techniques for this purpose are presented in the sequel. The first is based on inverting the sign of the confidence value for some bits. The second technique consists of coset projections based on invariant permutations of the code.

Lemma 4: An equivalent rule for ML soft-decision decoding of code C is $\min_{c \in C} \Sigma(c_i + g_i)\tilde{\mu}_i$, where $\tilde{\mu}_i \triangleq (-1)^{g_i}\mu_i$ and g is a binary n-tuple.

Proof: Recall the ML decoding rule (1) which implies

$$\begin{aligned} \max_{c \in C} \sum (-1)^{c_i} \mu_i \quad &\Leftrightarrow \quad \max_{c \in C} \sum (+1)^{c_i + g_i + g_i} \mu_i \\ &\Leftrightarrow \quad \max_{c \in C} \sum (-1)^{c_i + g_i} \tilde{\mu}_i \\ &\Leftrightarrow \quad \min_{c \in C} \sum (1 - (-1)^{c_i + g_i}) \tilde{\mu}_i \\ &\Leftrightarrow \quad \min_{c \in C} \sum (c_i + g_i) \tilde{\mu}_i. \end{aligned}$$ □

Let u denote the binary n-tuple obtained from μ by hard decision. Clearly (2) is a special case of Lemma 4 when $g = u$. Let $g = f + u$ for any binary n-tuple f. Then

$$\tilde{\mu}_i = (-1)^{f_i} |\mu_i|,$$

and the rule of Lemma 4 becomes

$$\min_{e \in K_f} \sum_{i=1}^{n} e_i \tilde{\mu}_i, \tag{11}$$

where $g \in K_f$. We have diverted the decoding problem of the received pattern in K_r to the equivalent problem of decoding in the coset K_f with modified confidence values vector $\tilde{\mu}$. However, $\tilde{\mu}$ is not non-negative in all coordinates, and this raises the question whether the techniques presented in previous sections, which assumed a positive sign of $|\mu|$, are still applicable. Fortunately the impact on those trellis techniques is negligible. In fact, we will show that the DB may still be eliminated and that only a minor adaptation is needed in the search algorithm. Denote by $\mathcal{I}_f$ the set of coordinates which correspond to I_f, and let $\mathcal{L}$ be the set of $\lfloor d/2 \rfloor$ coordinates corresponding to the least among $\{|\mu_i|\}$.

Theorem 5: If $\mathcal{I}_f \subseteq \mathcal{L}$ then any error pattern $e \in K_f$ with confidence values $\tilde{\mu}$ such that I_e forms a linearly dependent set may be eliminated.

Proof: Let $\bar{\mathscr{I}}_f$ be the complement set of $\mathscr{I}_f$. The ML rule (11) is equivalent to

$$\min_{e \in K_{\bar{f}}} \left\{ \sum_{i \in \bar{\mathscr{I}}_f} e_i |\mu_i| - \sum_{i \in \mathscr{I}_f} e_i |\mu_i| \right\}. \tag{12}$$

Consider an error pattern $e \in K_{\bar{f}}$ such that I_e forms a linearly dependent set. Denote by I_e^0, $I_e^0 \subseteq I_e$, a set of columns whose binary sum is $\mathbf{0}$. This set corresponds to a codeword, and therefore $|I_e^0| \geq d$. Suppose η columns from I_e^0 correspond to coordinates which are included in $\mathscr{L}$, $\eta \leq \lfloor d/2 \rfloor$. Hence, there are at least $\lceil d/2 \rceil$ coordinates from the complement set of $\mathscr{L}$ which correspond to columns in I_e^0. By definition, the confidence values related to $\mathscr{L}$ are minimal, and consequently the total sum of confidence values related to I_e^0 is non-negative. We conclude that the error pattern in $K_{\bar{f}}$, defined by $I_e - I_e^0$, must be at least as good as e, with respect to (12), and e may be eliminated. □

Corollary 1: All the results of Section III, referring to DB and generalized DB, hold also for $K_{\bar{f}}$ with confidence values $\tilde{\mu}$.

The search algorithm in Section IV requires only a slight adaptation due to the sign of $\tilde{\mu}$. Clearly the manipulations involved with zero-labeled branches and trivial branches are still applicable. The same applies also for the structure of Fig. 4(b). The only change concerns structures of the type in Fig. 4(a). In fact, such structures located between V_{i-1} and V_i may be processed with three operations instead of four only if $\tilde{\mu}_i$ is non-negative. This is not a significant loss in high-rate codes since then $\lfloor d/2 \rfloor \ll n$.

The method of diverting the decoding problem from one coset to another by inverting the sign of $\eta \leq \lfloor d/2 \rfloor$ confidence values from $\mathscr{L}$ will be referred as *sign inversion transformation* (SIT). This method is in fact compatible with the rest of the decoder principles. The computational cost, due to the partial sorting of η values, is (see also Section III)

$$n - 1 + (\eta - 1)(\lceil \log_2 n \rceil - 1). \tag{13}$$

The second type of coset projections is based on the following simple observation. Any pattern may be permuted, say by permutation P, and then decoded in the equivalent permuted code. In terms of the ML rule (2), $\min e|\mu|^t \Leftrightarrow \min (eP)(|\mu|P)^t$. Suppose $P \in \mathrm{AUT}(C)$, namely, P is an invariant permutation of C. Then the set K_rP, composed of patterns eP for $e \in K_r$, is also a coset of C [18, p. 237]. Recall from Section II that specific permutation is initially selected in order to reduce the minimal trellis size. Hence, we may wish to exploit now invariant permutation of an equivalent code of C, denoted by $\tilde{C}$. For example, it may be preferable to use cyclic permutation of a cyclic code $\tilde{C}$ rather than invariant permutations (if they exist at all) of C. Denote by Q the permutation by which $\tilde{C}$ is obtained from C. K_rQA designates a coset of $\tilde{C}$ in the *class* of K_rQ, where $A \in \mathrm{AUT}(\tilde{C})$. In order to return to the original code C, we perform the inverse permutation Q^t, thus $P = QAQ^t$. Let S and S' denote the syndromes of e and $eQAQ^t$, respectively. Hence, $S' = H(eQAQ^t)^t = HQQ^t(eQAQ^t)^t = HQ(eQA)^t = FHe^t = FS$, where F is a linear transformation defined by $FHQ = HQA^t$. Notice the principal distinction between Q and A: Q (or Q^t) is used to change the code into an equivalent code. However, A actually maps the error pattern into another one, in another coset, within the same code.

In summary, the strategy of selecting the coset representatives is as follows. A proper equivalent code $\tilde{C}$ is composed, and the permutations for projecting the coset representative are applied. A minimal set of cosets, with low decoding complexity, which spans the complete set of cosets is determined.

The invariant permutation technique will be demonstrated for two families of codes. The first includes the extended primitive BCH codes. Consider a primitive BCH code over $GF(2^m)$, and the correspondence between its coordinates and elements X_i in $GF(2^m)$, $i = 1, 2, \cdots, n$. It is well known that the extended BCH codes are invariant under the *affine permutation* $X_i' \leftarrow aX_i + b$, where $a, b \in GF(2^m)$, $a \neq 0$. Let σ be a primitive field element. It can be shown [18, p. 236] that

$$S_i' = \sum_{j=0}^{i} \binom{i}{j} a^j b^{i-j} S_j, \tag{14}$$

where $S_i = e(\sigma^i)$, $e(x)$ denotes the polynomial with coefficients e_j, and S_i' is the corresponding element for the permuted pattern e'.

Cyclic codes with odd length are also invariant under the conjugation $X_i'' \leftarrow X_i^{2^l}$ [14]. Clearly, the extended cyclic codes are also invariant under this permutation, where the checksum bit corresponding to $X_n = 0$ remains in its position. The new syndrome is

$$S_i'' = S_i^{2^l}. \tag{15}$$

Denote by $\mathscr{C}_i$ the *cyclotomic coset* which consists of the integer i and its conjugates modulus n. Let $\mathscr{C}(C)$ denote the set of cyclotomic cosets which correspond to the *minimal polynomials* $\{g_i(x)\}_{i \in \Omega}$ in the *generator polynomial* $g(x) = \prod g_i(x)$ of C, i.e., $\mathscr{C}(C) = \{\mathscr{C}_i : i \in \Omega\}$.

The procedure of selecting the coset representatives will be exhibited for the simple case $\mathscr{C}(C) = \{\mathscr{C}_0, \mathscr{C}_1\}$ (a more complicated case is detailed in Appendix A). From (14), setting $b = 0$ yields $S_0' = S_0$, $S_1' = aS_1$. If $S_1 \neq 0$, a may be chosen such that $S_1' = 1$. If $S_1 = 0$, then $S_1' = 0$. Hence, three cosets are sufficient for representing any extended Hamming code (we count only nonzero syndromes). Alternatively, by using also the SIT technique, we may determine the checksum bit for any received pattern u to be either 0 or 1. Thus we have managed to represent any extended Hamming code by a single coset!

The results for a few codes are listed in Table I, where CR_1 and CR_2 are the number of coset representatives with or without using SIT, respectively. CT is the total number of error cosets, i.e., $CT = 2^{n-k} - 1$.

TABLE I
COSET REPRESENTATIVES FOR EXTENDED PRIMITIVE BCH CODES

Code	n	k	Roots	CT	CR_1	CR_2
Hamming	2^m	$2^m - 1 - m$	0,1	$2^{m+1} - 1$	1	3
BCH	16	7	0,1,3	511	7	23
BCH	32	21	0,1,3	2047	3	19
BCH	64	51	0,1,3	8191	15	55
BCH	128	113	0,1,3	32767	3	43
BCH	256	239	0,1,3	131071	37	143
BCH	64	45	0,1,3,5	524287	55	929
BCH	128	106	0,1,3,5	4194303	61	2535

The second family of codes (which includes the first) consists of cyclic and extended cyclic codes. Recall the correspondence between elements $X_i \triangleq \sigma^i$ and the coordinates i of a cyclic (n, k) code over $GF(2^m)$, $i = 1, 2, \cdots, n$, where σ is a primitive nth root of unity. With this notation, the cyclic permutation of y positions may be expressed by $X_i' \leftarrow \sigma^y X_i$. Then $S_i' = \sum X_j'^i e(X_j) = \sum (\sigma^y X_j)^i e(X_j) = \sigma^{yi} \sum X_j^i e(X_j)$, thus

$$S_i' = \sigma^{yi} S_i. \tag{16}$$

The conjugation permutation and Eq. (15) exist for any cyclic code with odd length n. A projection scheme is established, motivated by a strategy similar to the one for the first family of codes. The process is elaborated in Appendix B for several types of cyclic codes, and a few examples are provided in Table II. Notice that in the case of extended cyclic codes, the checksum bit is not reflected by the cyclic or conjugate permutation.

TABLE II
COSET REPRESENTATIVES FOR CYCLIC AND EXTENDED CYCLIC CODES

Code	n	k	Roots	CT	CR_1	CR_2
Hamming	$2^m - 1$	$2^m - 1 - m$	1	$2^m - 1$		1
Exp. Hamm.	$2^m - 1$	$2^m - 2 - m$	0,1	$2^{m+1} - 1$	1	3
BCH	17	9	1	255		35
BCH	15	7	1,3	255		11
BCH	21	12	1,3	511		27
BCH	31	21	1,3	1023		9
BCH	35	20	1,5	32767		703
BCH	35	28	5,7	127		25
BCH	63	55	1,21	255		15
BCH	127	113	1,3	16383		21
Ext. BCH	22	12	0,1,3	1023	27	55
BCH	63	53	0,1,9	1023	5	11

TABLE III
WORST CASE DECODING COMPLEXITY

Code	n	k	d	GDB	SIT	CR	N_W	N_O	N
Hamming	7	4	3	No	No	1	14.7	3.5[a]	3
BCH	15	10	4	Yes	Yes	2	63.5	16.5[b]	9
Cyclic	21	15	4	No	Yes	4	136		22
BCH	35	28	4	No	Yes	25	311		38
R-R-C[c]	60	53	4	No	No	127	345		65
Exp. Hamm.	63	56	4	No	Yes	1	368		62
Exp. Hamm.	127	119	4	No	Yes	1	727		127
Ext. Hamm.	128	120	4	No	Yes	1	727		127
Ext. Hamm.	256	247	4	No	Yes	1	1490		256
BCH	31	20	6	Yes	Yes	9	3277		859
BCH	33	22	6	Yes	Yes	107	3537		504

[a] From [5].
[b] Obtained from a decoder of [7] for the Hamming (31, 26) code.
[c] Repeated root cyclic code—See [9] for a proper permutation for these codes.

VI. RESULTS AND CONCLUSIONS

A systematic design of a soft-decision decoder for block codes is presented. First a certain permutation is determined which aims to shrink the trellis size. Then the coset trellises are constructed and punctured, and the computational complexity is calculated for each coset according to the improved search algorithm. This is a one-time effort which may all be done by a computer program. The decoding complexity involved with each coset, as a consequence of the specific puncturing and projection techniques used, is employed to determine the preferable set of coset representatives. Many efficient decoders have been constructed by this design, some of which are reported in Table III. The optional puncturing of generalized DB and the use of the SIT technique are indicated respectively under *GDB* and *SIT*. The consequent number of coset representatives, which reflects the storage complexity, is denoted by *CR*. The normalized computational complexity (per information bit), denoted by N, is compared to the general Wolf [1] trellis-based algorithm (N_W) and to other best-known decoding schemes (N_O), when applicable.

Suboptimal decoding algorithms such as in [14], [19] are based on accounting only error patterns with bounded Hamming weight. In the puncturing process, Algorithm A may be utilized to eliminate patterns with any prescribed weight from the trellis, consequently reducing the resulting decoding complexity. Hence, the proposed optimal decoder design may be straightforwardly adapted for more practical bounded-distance decoding schemes. A subsequent study could be to investigate the trade-off between performance and complexity of such a decoder.

Finally we remark that some of the techniques presented in this paper may be independently utilized in other related applications. For example, the coset projection methods of Section V may drastically reduce the storage complexity of any syndrome-based decoder.

APPENDIX A

The invariant permutations technique for coset projections is demonstrated in this appendix for the extended primitive BCH codes characterized by $\mathscr{C}(C) = \{\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_3\}$. In a field with characteristic 2, $S_2 = S_1^2$. Consequently, (14) implies $S_0' = S_0$, $S_1' = aS_1 + bS_0$, and $S_3' = a^3 S_3 + a^2 b S_1^2 + ab^2 S_1 + b^3 S_0$. Choose $b = 0$. A proper selection of a sets S_1' to either 0 or 1: If $S_1 \neq 0$ then S_3' may be transformed to the leader of its cyclotomic coset by employing the conjugation permutation. Otherwise, when $S_1 = 0$, $S_1' = 0$, and also $S_3' = a^3 S_3$. In the general case, the solution for the cubic equation $a^3 = S_3' S_3^{-1}$ is not guaranteed.

Lemma 5: Let σ be a primitive nth root of unity. There is a solution to the equation $x^i = \beta$ in the multiplicative group $\langle \sigma \rangle$, generated by σ, for all $\beta \in F_2$, if and only if F_2 is a subset of $F_1 \triangleq \langle \sigma^i \rangle$.

Proof: Suppose there is a solution $x = \sigma^{z_\beta}$ to the equation for all $\beta \in F_2$, i.e., $(\sigma^{z_\beta})^i = \beta$. Then $(\sigma^i)^{z_\beta} = \beta$, which proves that $F_2 \subseteq F_1$. By following the same arguments backwards the proof is completed. □

In our case $S_3' = a^3 S_3$, $n = 2^m - 1$, and (assuming $S_3' = 1$) F_2 consists of the inverse elements of span$\{\sigma^0, \sigma^3, \cdots, \sigma^{3(n-1)}\}$. Suppose there are λ cyclotomic cosets in $GF(2^m)$. Lemma 5 implies that if $F_2 \subseteq \langle \sigma^3 \rangle$, e.g., when $(n,3) = 1$, then S_3' may always be set to either 0 or 1. Consequently, at most $2\lambda + 5$ coset representatives are required. However, if no solution exists, then, with conjugation, at most $4\lambda + 3$ are needed.

Alternatively, we may use first the SIT technique, forcing $S_0 = 0$. If $S_3 = 0$, choose $b = 0$. Hence $S_3' = 0$ and S_1' is either 0 or 1. Otherwise, when $S_3 \neq 0$, if $S_1 = 0$, then $S_1' = 0$ and $S_3' = a^3 S_3$. Again, S_3' is set to 1 unless no solution exists and the conjugation permutation transfers S_3' to its coset leader.

If both S_1 and S_3 are nonzero, a is determined such that S_1' is set to 1. The following equation is obtained for S_3': $S_3' = S_1^{-3} S_3 + b + b^2$. There is a solution to this quadratic equation if $\mathrm{tr}(S_3' + S_1^{-3} S_3) = 0$ [14]. A known identity is [18, p. 116] $\mathrm{tr}(\beta_1 + \beta_2) = \mathrm{tr}(\beta_1) + \mathrm{tr}(\beta_2)$ for any field elements β_1, β_2. Hence we may find an appropriate b such that S_3' is either 0 or 1, according to $\mathrm{tr}(S_1^{-3} S_3)$. We conclude that the total number of representatives required with SIT is either 3 or at most $2 + \lambda$, in compliance with the inherent code properties.

This technique can be further applied for more complicated cases such as codes with $\mathscr{C}(C) = \{\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_3, \mathscr{C}_5\}$ and so on.

Appendix B

The invariant permutations technique for coset projections is demonstrated here for the family of cyclic (and extended cyclic) codes. The method is exhibited for a few cases.

1) $\mathscr{C}(C) = \{\mathscr{C}_i\}$, $i = 0, 1, \cdots, n$: It follows from (16) that $S_i' = \sigma^{iy} S_i$. Here $F_1 = \langle \sigma^i \rangle$ and F_2 consists of the inverse elements of span$\{\sigma^0, \sigma^i, \cdots, \sigma^{(n-1)i}\}$. Assume that an error occurs, so that $S_i \neq 0$. If there is a solution to the equation above, only one coset representative is required, say $S_1' = 1$. Otherwise, by conjugation (assuming n is odd), at most λ cosets suffice. Note that λ is generally the number of cyclotomic cosets in the splitting field of the code, $GF(2^m)$.

2) $\mathscr{C}(C) = \{\mathscr{C}_0, \mathscr{C}_i\}$, $i = 1, 2, \cdots, n$: Eq. (16) implies that $S_0' = S_0$, $S_i' = \sigma^{iy} S_i$. F_1, F_2 are defined as in case 1. If there is a solution to the second equation, three coset representatives suffice. Otherwise, by conjugation (assuming n is odd), at most $2\lambda + 1$ representatives are required. We may alternatively use first the SIT technique, assigning 0 or 1 to S_0'. If $S_0' = 0$, then either one or λ representatives suffice, depending on the existence of a solution to the second equation. Otherwise, two or no more than $\lambda + 1$ representatives are needed, respectively.

3) $\mathscr{C}(C) = \{\mathscr{C}_i, \mathscr{C}_j\}$, $i, j = 1, 2, \cdots, n$: Again, $S_i' = \sigma^{iy} S_i$, $S_j' = \sigma^{jy} S_j$. Assume that the syndrome is not zero. If exactly one of S_i and S_j is zero, then if there is a solution to the corresponding equation of the other element, one coset suffices. If no solution exists then λ cosets are required at the most. If both S_i and S_j are nonzero, then either of them may be set to 1 or to a cyclotomic coset leader, resulting in one or at most λ values, respectively. The other element can be either transformed to a cyclotomic coset leader or cannot be transformed at all. We conclude that the total number of representatives is at most $\lambda + 2$ in case both solutions exist, and up to $\lambda(2^m + 1)$ in the converse case.

References

[1] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76–80, 1978.

[2] J. H. Conway and N. J. A. Sloane, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 1, pp. 41–50, 1986.

[3] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 355–364, 1986.

[4] G. D. Forney, Jr., "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-34, no. 5, pp. 1152–1187, 1988.

[5] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 963–975, 1989.

[6] A. Vardy and Y. Be'ery, "More efficient soft decoding of the Golay codes," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pp. 667–672, 1991.

[7] J. Snyders, "Reduced lists of error patterns for maximum likelihood soft decoding," *IEEE Trans. Inform. Theory*, vol. 37, no. 4, pp. 1194–1200, 1991.

[8] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 1049–1053, 1988.

[9] Y. Berger and Y. Be'ery, "Bounds on the trellis size of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 1, pp. 203–209, 1993.

[10] V. K. Wei and K. Yang, "On the generalized Hamming weights of squaring construction and product codes," preprint.

[11] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the state complexity of trellis diagrams for Reed–Muller codes and their supercodes," in *Proc. 14th Symp. Inform. Theory and its Appl.*, (Japan), Dec. 1991.

[12] G. D. Forney, Jr. and M. D. Trott, "The dynamics of linear codes over groups: State spaces, trellis diagrams and canonical forms," preprint.

[13] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, 1967.

[14] E. R. Berlekamp, "The construction of fast, high-rate, soft-decision block decoders," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 3, pp. 372–377, 1983.

[15] A. Vardy and Y. Be'ery, "On the problem of finding zero-concurring codewords," *IEEE Trans. Inform. Theory*, vol. 37, no. 1, pp. 180–187, 1991.

[16] A. Vardy, J. Snyders, and Y. Be'ery, "Bounds on the dimension of codes and subcodes with prescribed contraction index," *Linear Algebra Appl.*, vol. 142, pp. 237–261, 1990.

[17] D. E. Knuth, *The Art of Computer Programming*, Vol. 3. Reading, MA: Addison-Wesley, 1973.

[18] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1977.

[19] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 2, pp. 170–182, 1972.