Maximum Likelihood Soft Decoding of Binary Block Codes and Decoders for the Golay Codes

JAKOV SNYDERS, MEMBER, IEEE, AND YAIR BE'ERY, MEMBER, IEEE

Abstract — Maximum likelihood soft decision decoding of linear block codes is addressed. A binary multiple-check generalization of the Wagner rule is presented, and two methods for its implementation, one of which resembles the suboptimal Forney-Chase algorithms, are described. Besides efficient soft decoding of small codes, the generalized rule enables utilization of subspaces of a wider variety than those recently applied (by Conway and Sloane, Be'ery and Snyders, and Forney) for coset decoding, thereby yielding maximum likelihood decoders with substantially reduced computational complexity for some larger binary codes. More sophisticated choice and exploitation of the structure of both a subspace and the coset representatives are demonstrated for the (24, 12) Golay code, yielding a computational gain factor of about 2 with respect to previous methods. A ternary single-check version of the Wagner rule is applied for efficient soft decoding of the (12, 6) ternary Golay code.

I. INTRODUCTION

MAXIMUM LIKELIHOOD soft decision decoding of block codes and lattices has recently gained renewed interest. Most notably, the algorithms derived in [1]-[4] are considerably more efficient than earlier results and implementable in practice for various small codes. The decoding procedures in the foregoing papers, are presented with the aid of different tools, namely tables [1], Hadamard transforms [2], and trellises [3], [4]. Nevertheless, they all rely on the same idea: decoding each coset of a subspace of a certain type, in combination with the Wagner rule [5], when appropriate.

The Wagner decoding rule, which applies to binary codes whose check matrix consists of a single all-ones row, is perhaps the oldest maximum likelihood soft decision decoding algorithm. It states that an entry-by-entry hard decision of the received word is to be followed, unless the number of 1 bits is already even, by inversion of the least reliable bit. Several suboptimal algorithms contain, or in a sense, generalize the concept of the Wagner rule. These include the algorithms of Silverman and Balser [5], Forney [6], and Chase [7] for soft decoding of binary linear block codes, Forney's sequence estimation algorithm [8] for partial response channels, and the fast block decoders of Berlekamp [9], [10]. Conway and Sloane [11] generalized the Wagner rule for decoding the checkerboard lattices $D_n$ (the lattice analogs of single parity-check codes), which they then applied [1] to maximum likelihood decoding of other lattices. Forney [4] applied a single-check ternary generalization of the Wagner rule for maximum likelihood decoding of the Leech lattice, based on a decoder for the (12, 6) Golay code. An application of the Wagner rule for maximum likelihood decoding of a nonlinear code was presented by Abbaszadeh and Rushforth [12].

A general coset-decoding principle for maximum likelihood decoding of both block and lattice codes is described in [1]. However, it was implemented [1]-[4], [12] with respect to subcodes that belong to a limited class, and the possibility of efficient decoding by employing several other types of subcodes, suggested in [1], has remained apparently unexplored. The subspaces utilized for decoding block codes are spanned by codewords that are, in the terminology of [2], either zero-concurring (i.e., the ones nowhere overlap) or concurrent (i.e., the ones nowhere overlap except in several positions, where ones of all codewords overlap). These subspaces are, in a sense, contractible to universal (i.e., zero parity-check) codes and single parity-check codes, respectively. (Contractibility, defined in Section IV, is the term used here for a generalization of geometric similarity [1].) Properties of the coset representatives of the code, relative to the subspace adopted, are exploited in [3] more intensively than in [1] and [2], and in a different manner. This is the basic reason for the differences in computational complexity given in [1]-[3] for comparable codes, particularly the (24, 12) and (23, 12) Golay codes (the ways of counting also differ slightly). The methods developed in [1]-[3], and in [4] for the ternary case, although useful for several codes, are not efficient enough for many other codes. They may be particularly inefficient for longer codes with reasonably high rates, where the dimension of the largest available subspace of the two types utilized is a rather small fraction of the dimension of the codes, and the numerous coset representatives lack an attractive structure.

In this paper a binary multiple-check generalization of the Wagner rule is presented, its implementation for codes
with a few parity bits is described, and for other codes its application within the framework of coset decoding is examined. In the latter case, the subspaces utilized are contractible to codes with $\lambda$ check bits, where $\lambda$ is allowed to exceed 1. Therefore, the class of codes for which efficient maximum likelihood decoders are derivable is augmented, and for some codes a further reduction of computational complexity is available. A fundamental component of a decoder based on this approach is a search scheme, whose complexity tends to grow rapidly with the increase of the number of check bits. An additional approach, of exploiting (more intensively than in [1]–[3]) the structure of both a subcode and the set of coset representatives, yields a rather efficient decoder for the (24,12) Golay code. Although demonstrated here for a particular code, and with subspace geometrically similar to a single-check code, the latter method is successfully applicable to other codes including those with $\lambda > 1$ checks, provided that the subcode and the coset representatives possess certain symmetries. We remark that the particularly symmetric structure of the (24,12) Golay code was exploited for efficient hard decision decoding by Conway [14] and, subsequently, by Pless [15]. A decoder for the (12,6) Golay code, more efficient than that of [4], is also derived. The general $q$-ary case is treated elsewhere [16]. Decoders for the Leech lattice, based on both single- and double-check ideas, and computationally less complex than those of [1], [3], [4], and [21] are derived in [17].

The following is an outline of the paper. A Wagner decoding rule generalized to multiple checks, Algorithm 1, and its implementation are presented in the next section. The complexity of decoding for a code with two parity bits turns out to be the same as for the original case where there is a single-check bit. Also, it is relatively small when the number of parity bits is three or four and, for some codes, even more. Decoders derived by this approach for the (7,4) and (15,11) Hamming and the (8,4) Reed-Muller codes are more efficient (in terms of real addition-equivalent operations) than previously known maximum likelihood decoders. Berlekamp’s decoder [10] performs a search that may be viewed as a suboptimal version of the exhaustive schemes discussed in Section II. An alternative implementation of Algorithm 1, based on an ordered list of confidence values, is considered in Section III. It resembles the suboptimal Forney [6] and Chase [7] algorithms. Ordering has certain advantages when the code is sufficiently long and contains at least four parity bits. In Section IV utilization of a subcode, in a fashion similar to the approach of [1]–[4], is considered. However, the procedure adopted, as formulated in Algorithm 2, uses specifically the generalized Wagner rule. Expressions that provide an idea of the computational complexity of this algorithm are presented. The actual complexity is sometimes much smaller, as seen, in particular, in Example 6. For each of the codes considered in the examples, computational gain was achieved by increasing $\lambda$, the number of checks, from zero or one to two or three. The complexities obtained are 132, 244, and 340 addition-equivalent operations for (15,7), (17,9), and (18,9) codes, respectively, significantly lower than any previously known. A decoder for the (24,12) Golay code, that exploits the structure of both the subcode and the coset code, is derived in Section V. Its complexity is 827 addition-equivalent operations at most, and about 683 operations on average, compared to 1351 operations in [3], at most 1551 operations in [2], and 1614 operations in [1]. It is also distinguished by its small memory-space requirement. Two decoders, corresponding to $\lambda = 0$ and $\lambda = 1$, for the (12,6) Golay code are presented in Section VI. The complexity of the more efficient one ($\lambda = 1$) is 656 addition-equivalent operations at most, compared to 1061 operations in [4].

II. A GENERALIZED WAGNER ALGORITHM

Let $C$ be an $(n, k, d)$ binary linear block code of length $n$, dimension $k$, and minimum distance $d$. Codewords of $C$ are transmitted through a memoryless channel characterized by transition probability densities (alternatively, probabilities) $p(v|j)$ where $j \in GF(2)$ and $v$ belongs to the output alphabet, usually the set of real numbers. Let $v = (v_0, v_1, \cdots, v_{n-1})$ be a word received at the channel output. Maximum likelihood soft decision decoding can be carried out by first making a symbol-by-symbol hard decision, thus obtaining some $n$-tuple $b$, and then changing the initial decisions by the error pattern $e$ for which the sum of the magnitudes of the log likelihood ratios corresponding to the nonzero entries of $e$ is minimum, among all the $n$-tuples of $b$ in the coset of $C$ that contains $b$.

To establish this claim formally, recall [2], [13] that maximum likelihood decoding consists of identifying a codeword $e = (e_0, e_1, \cdots, e_{n-1}) \in C$ which maximizes the following expression

$$M(e) = \sum_{j=0}^{n-1} (-1)^j \mu(e_j)$$

where $\mu(v) = \log [p(v)/p(e)]$. Obviously,

$$\max_{e^* \in GF(2)^n} M(e^*) = \sum_{i=0}^{n-1} \mu(e_i),$$

and the (unique, unless some members of $\mu(e_i)$ are zero) $n$-tuple $e^*$, denoted by $b$, for which $M(e^*)$ attains the maximum, is given by

$$b = (\text{sgn} \mu(e_0), \text{sgn} \mu(e_1), \cdots, \text{sgn} \mu(e_{n-1}))$$

where

$$\text{sgn}(x) = \begin{cases} 0, & x \geq 0 \\ 1, & x < 0. \end{cases}$$

Consequently, maximization of (1) and minimization of the following expression

$$T(e) = \sum_{i=0}^{n-1} \left[ (-1)^{\text{sgn} \mu(e_i)} - (-1)^i \right] \mu(e_i),$$

both with respect to all $e \in C$, are equivalent. By writing $c_i = \text{sgn} \mu(e_i) \Phi e_i$, where $\Phi$ stands for summation mod-
ulo 2, we obtain
\begin{equation}
T(e) = \sum_{i=0}^{n-1} (1 - (-1)^e_i) \mu(e_i) = 2 \sum_{i=0}^{n-1} e_i \mu(e_i).
\end{equation}

Of course, \( e = (e_0, e_1, \ldots, e_{n-1}) \) and \( b \) belong to the same coset of \( C \).

Let \( h_i, i = 0, 1, \ldots, n-1 \) be the columns of a parity check matrix \( H \) of \( C \), and let the syndrome \( z \) associated with \( b \) be defined as the column vector
\begin{equation}
z = Hb^t
\end{equation}
where the superscript \( t \) denotes transposition. The decoding procedure may be rephrased as follows: complement \( b \) over a set of locations \( L \subset \{0, 1, \ldots, n-1\} \), for which \( \sum_{i \in L} h_i (a \bmod 2) \sum_{i \in L} \mu(e_i) \) is minimum. Notice that, for a binary symmetric channel, \( \mu(e) \) is equivalent to \( (-1)^e \) and, consequently, the Hamming weight of \( e \) is minimized, as expected.

Of course, \( L = \emptyset \) if \( z = 0 \). Assuming that \( z \neq 0 \), it is sufficient to consider only those location sets for which the corresponding columns of \( H \) are linearly independent. This follows by observing that the optimal location set \( L \) is unique and \( \{h_i; i \in L\} \) are linearly independent, provided that \( \mu(e_i) \neq 0 \) for all \( i \) and a tie, which (although unlikely in true soft decision decoding) otherwise may occur, is resolvable in favor of independence. Consequently, \( |L| \leq n-k \), where \( \cdot \) with respect to a set stands for cardinality. Also, the position of at most one of a set of identical columns of \( H \) belongs to \( L \). That position is identifiable by minimizing the confidence value, \( \mu(e_i) \), over the set. More explicitly, denote \( m = \{h_i; i = 0, 1, \ldots, n-1\} \) and write \( \{0, 1, \ldots, n-1\} = \bigcup_{j=1}^m L_j \) where \( L_j \) are specified by the requirement that \( |\{h_i; i \in L_j\}| = 1 \) for each \( j = 1, 2, \ldots, m \). Let
\begin{equation}
\psi_j = \min_{i \in L_j} \mu(e_i), \quad j = 1, 2, \ldots, m.
\end{equation}

We shall summarize now the maximum likelihood decoding rule.

Algorithm 1:

a) Perform hard decisions \( b \) and evaluate the corresponding syndrome \( z \) given, respectively, by (2) and (5). Accept \( b \) if \( z = 0 \); otherwise go to step b).

b) Compute the minimum confidence value \( \psi_j; j = 1, 2, \ldots, m \) (6) over each set of identical columns of \( H \). Store these values, together with the locations where they are attained.

c) Among all the sets of linearly independent columns of \( H \) whose members add up to \( z \) (modulo 2), find the (usually unique) one for which the sum of confidence values is minimum.

d) Complement the bits of \( b \) associated with the set found.

Consider a code in which a single parity bit checks each of the codewords, i.e., \( H \) is an all-ones row. Then Algorithm 1 reduces to the Wagner decoding rule [5], which requires that a bit of \( b \) with lowest confidence value be complemented if \( z = 1 \). It is noteworthy that the decoding rule for this extremely trivial case is the origin of maximum likelihood decoding algorithms that are efficiently applicable for even high rate, though short, codes [1], [2], [3], [12]. An algorithm that employs a slightly more general type of subspace, spanned by the so-called partially occurring codewords, was also described in [2, p. 363]; it may be regarded as an application of Algorithm 1 to the case where \( H \) is a single row with possibly some zero entries (for an example, see (22), (23)). In the remainder of this section we shall discuss several other simple cases that are, however, interesting, partly in their own right but mostly as a means of carrying out Algorithm 2 of Section IV.

First let \( n-k = 2 \). To consider the computationally most complex situation, assume that all of \( (0 \ 1)^t \), \( (1 \ 0)^t \) and \( (1 \ 1)^t \), but not \( 0 \), are columns of \( H \). Then, unless \( z = 0 \), three minimizations of step b) require altogether \( n-3 \) addition-equivalent operations, and step c) is performed by comparing the confidence value corresponding to the column that equals the syndrome with the sum of the confidence values corresponding to the other two columns. Thus the complexity totals \( n-1 \) addition-equivalent operations, as for the case \( n-k = 1 \) with no zero entry in \( H \). More precisely, the case \( n-k = 2 \) is slightly more complex than the other one, due to the separation into groups and the addressing of memory involved. In computational complexity evaluations, we shall, however, ignore those kinds of operations, as well as the checking of logical conditions and modulo 2 additions, since the latter are also attributable to memory. Notice that decoding with the aid of tables [1] or trellises [3], [4] also necessitates memory, which is usually excluded from complexity evaluations.

For the case \( n-k = 3 \), assuming that every binary 3-tuple except \( 0 \) is a column of \( H \) and \( z \neq 0 \), the complexity of step b) is \( n-7 \). For step c), first the six 3-tuples which differ from \( z \) are grouped into three distinct pairs. By a pair we mean a set of two linearly independent columns of \( H \) whose sum is the syndrome \( z \). Similarly, a set of three linearly independent columns of \( H \) that add up to \( z \) is named a triplet, etc. Let \( A, B, \) and \( C \) be the pairs mentioned. The triplets may be formed by picking arbitrarily one vector from each of the pairs \( A \) and \( B \) and adjusting to them a member of \( C \). A change of selection from either \( A \) or \( B \) induces a change in the vector picked from \( C \). This yields the four triplets. The complexity of decoding is thus \( (n-7) + 3 \cdot 1 + 4 \cdot 2 + 7 = n + 11 \) real additions, including the complexity 7 of the final minimization.

A modest decrease in complexity results by finding the sum over the best triplet as follows. Select from the confidence values over each pair the smaller one and, in case the vectors associated with these values form a triplet, add them. Otherwise, find the pair over which the confidence values differ the least, then add the larger confidence value over that pair to the sum of the smaller confidence values over the other two pairs. This procedure requires, respectively, either \( 3 + 2 = 5 \) or \( 3 + 2 + 2 = 7 \) addition-equivalent operations (three for minimizing over the pairs, two for
finding the minimal difference, if necessary, and two more for summation). In both cases the final minimization is over five numbers, namely, the sum of confidence values over each of the three pairs, the best triplet and the syndrome. Hence the complexity is either \( (n - 7) + 3 \cdot 1 + 5 + 4 = n + 5 \) or \( n + 7 \) additions. Thus the average is about \( n + 6 \). We remark that the average complexity of Algorithm 1 makes sense in particular within the framework of Algorithm 2.

**Example 1:** The (7,4) Hamming code is decoded by at most \( n + 7 = 14 \) additions when the latter method is applied (and by \( n + 11 = 18 \) additions when the more straightforward approach is followed). Forney’s (8,4) decoder [3], adjusted to the (7,4) code, performs 21 additions. Wolf’s method [18] requires 57 additions [2].

**Example 2:** Consider the (8,4) Reed–Muller (or extended Hamming) code equipped with the following check matrix:

\[
H = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

When the topmost entry of \( z \) is 1, the sum of confidence values over each of the 7 \( = 1 \) trips is \( 7 \) and over \( \{ z \} \) have to be compared. The eight sets mentioned are related as shown in Fig. 1. Each node represents a column of \( H \) and each branch, consisting of a root and two other nodes, represents a triplet. The numbering in Fig. 1 is for the case \( z = (1 \ 1 \ 1 \ 1) \)' with each column of \( H \) designated by its location number (the assignment of columns to the branches, for a given \( z \), is not unique). To perform the minimization efficiently, first sum the confidence values corresponding to the two upper nodes of each branch, then find, in each tree separately, the minimum of these sums and add it to the confidence value of the root. This procedure identifies the best branch in each tree, while requiring only \( 7 + 4 + 3 = 14 \) additions. The final minimization is over four numbers. Thus the complexity is \( 14 + 3 = 17 \). Minimization over the four triplets represented by the two two-branch trees of Fig. 1 is also possible by applying the foregoing shortcut, as described for the case \( n - k = 3 \), to the following sets of two columns: \( \{ 2, 5 \} \), \( \{ 3, 4 \} \), and \( \{ 1, 6 \} \). Accordingly, the decoding is performed by either \( 5 + 6 + 2 = 13 \) or 15 additions, or only about 14 additions on the average.

![Fig. 1. Syndrome and seven triplets for case \( z = (1 \ x \ x \ x) \)' (with numeration corresponding to \( z = (1 \ 1 \ 1 \ 1) \)').](image)

If the topmost entry of a nonzero \( z \) is 0, then minimization over four pairs and \( 8 \cdot 6 \cdot 4 \cdot 1/4! = 8 \) quartets is due. The four pairs may be used to generate \( 2^3 \cdot 1 = 8 \) quartets. Accordingly, to minimize the sum of confidence values over the eight quartets, first find the vector associated with the less reliable bit in each of the four pairs. If these vectors do not form a quartet, then replace a single vector by its companion in that pair over which the confidence values differ the least. Now sum the confidence values over the quartet thus obtained. This procedure requires either \( 4 + 3 = 7 \) or \( 4 + 3 = 10 \) additions. Hence the complexity for this case totals either \( 4 \cdot 1 + 7 + 4 = 15 \) or \( 15 + 3 = 18 \) additions, including the final minimization over five numbers. Thus the complexity of decoding the (8,4) code never exceeds 18, and it is about 16 on the average. The trellis-based decoder of [3] requires 23 additions. The method of [18] may require up to 73 additions.

It is easily verified that, when all binary \( (n - k) \)-tuples are available, the number of distinct sets of \( r \) linearly independent vectors that add up to \( z \in GF(2)^{n-k} \), \( z + 0 \), is given by

\[
\frac{1}{r!} \prod_{i=1}^{r-1} (2^{n-k} - 2^i), \quad 2 \leq r \leq n - k.
\]

Now let \( n - k = 4 \), and assume that every binary 4-tuple \( \mathbf{0} \) is a column of \( H \) and \( z \neq \mathbf{0} \). The complexity of step b) is \( n - 15 \) and, by (7), there are seven pairs, 28 triplets, and 56 quartets. Minimization of the sum of confidence values over all these sets and \( \{ z \} \) by the straightforward approach requires 322 real additions. Accordingly, the complexity totals \( n + 307 \). However, the seven pairs can be grouped into \( 7 \cdot 6 \cdot 1/3! = 7 \) distinct (but nondisjoint) sets of three pairs each, such that each set provides \( 2^2 \cdot 1 = 4 \) triplets. Also, the 56 quartets are obtainable by forming \( 7 \cdot 6 \cdot 4 \cdot 1/4! = 7 \) sets of four pairs each, in a way that no set of three pairs used for producing triplets is a subset of four pairs; furthermore, each such set of four pairs yields \( 2^3 \cdot 1 = 8 \) quartets. Further explanations are deferred to Appendix I, where it is also shown that, by shortcuts of the type described earlier, decoding can be carried out in at most \( n + 68 \) additions.

**Example 3:** The (15,11) Hamming code is decoded in at most \( n + 68 = 83 \) additions. The previous best result is 221, obtainable with Forney’s decoder for the (16,11) code [3]. According to the method of counting as explained in Section IV, the lowest complexity achieved in [2] is 254, while Wolf’s method [18] requires 409 additions.

The number of sets given by (7) grows very rapidly with \( n - k \). For example, the number of quintets alone for the case \( n - k = 5 \) is 2688, and although these quintets are derivable from sets of five pairs each, there are \( 15 \cdot 14 \cdot 12 \cdot 8 \cdot 1/5! = 168 \) such sets. Consequently, for a code with a large number of check bits, even when not all the \((n - k)\) tuples are columns of \( H \), the previously described methods of implementation are prohibitive. In that case a suboptimal procedure may be formed by excluding from consideration, for example, all the error patterns whose Hamming weight exceeds some \( w < n - k \). The lower edge of the error-weight spectrum raises less concern. In particular, in each coset the vectors with weight \((d + 1)/2\) have disjoint support, and there is at most a single vector whose weight is smaller than \((d + 1)/2\).
Let C be the (128,106,8) BCH code. A decoder for C devised by Berlekamp [10] corrects all the error patterns of weight 4 or less and those of weight 5 whose support contains the location of the least reliable bit. To do this, first an even-weight hard decision vector \( \mathbf{b} \) is formed, by complementing the least reliable bit of \( \mathbf{b} \) if \( \mathbf{b} \) has odd weight. All the vectors in the coset of \( C \) to which \( \mathbf{b} \) belongs have even weight, because \( C \) is an extended code. Hence at most 128/4 = 32 quartets and one pair have to be scored. Identification of the actually existing error patterns, for any given coset, was efficiently executed in [10] by a syndrome transformation.

III. SEARCH SCHEMES BASED ON ORDERED LIST

For a large high-rate code, an alternative approach to decoding begins with ordering the columns of \( H \) according to nondecreasing confidence value of the associated bits. Obviously, at least some members of the optimal set are confined, with high probability, to the first few locations of the list.

A nonexhaustive search method based on this notion is due to Forney [6]. It consists of erasures-and-errors decoding of several versions of \( b \), and subsequent selection of the best codeword thus obtained. The versions of \( b \) are obtained by erasing \( \eta \) bits that are less reliable than the remaining bits, where \( \eta \) is all of the following: \( \eta = 0, 2, 4, \cdots, d - 1 \) for odd \( d \) while \( \eta = 0, 1, 3, \cdots, d - 1 \) for even \( d \). In a family of suboptimal algorithms presented by Chase [7] \( \eta \) least reliable bits of \( b \) are complemented and the resulting vector is subsequently decoded; in one version \( \eta \) takes on the previously mentioned values, whereas \( \eta \) is all of \( 0, 1, 2, \cdots, [d/2] \) in another version.

An exhaustive search scheme, based on an ordered list of columns of \( H \), is considered here. It has relatively low complexity for moderate values of \( n - k \). However, it is inefficient when \( n \) is too small. For example, for the (7,4) Hamming and the (8,4) Reed–Muller codes, the complexity of sorting alone may almost equal the complexity of decoding obtained earlier. Prior to ordering, step b) of Algorithm 1 is performed.

Let us reexamine step c) for the case \( n - k = 4 \) with 15 pairwise-different nonzero columns. The complexity of sorting is at most 42 [19]. Let the ordered list of columns be identified by location labels 1,2,\cdots, 15, where 1 designates the column associated with the least reliable bit. If the syndrome \( z \) is labeled either 1 or 2, the search is complete. If the location label \( t \) of \( z \) satisfies \( t \geq 3 \), then it is enough to seek those pairs, triplets, and quartets (if any) whose labels do not exceed \( t - 1 \). If \( t \leq 8 \), then the number of such sets is quite small. Turning to the case \( t > 8 \), we suggest the following. Let \( i \) and \( j \), \( i < j \), be the location labels of that pair for which the larger label is minimum. Obviously, \( j \leq 8 \). Now sum the confidence values over those pairs whose labels are not larger than \( t - 1 \), and over those triplets and quartets where either all labels are less than \( j \) or one of the labels exceeds \( j - 1 \), but not \( t - 1 \), and all other labels are less than \( i \).

According to this scheme, the most complex situation occurs when \( i = 7 \), \( j = 8 \), and \( t = 15 \). Then the sum of confidence values over all the seven pairs must be computed. In addition, as shown in Appendix II, at most 15 triplets and 16 quartets have to be scored and, consequently, with a straightforward evaluation method applied to them, the worst case complexity totals \( n + 150 \). However, the number of required additions is frequently smaller even for the case \( i = 7 \), \( j = 8 \), and \( t = 15 \): it can be as small as \( n + 106 \), and even less if the sorting is performed with fewer than 42 operations. Furthermore, the complexity is obviously at most \( (n - 15) + 42 = n + 27 \) when \( t = 1, 2 \). Also, it is at most \( n + 29 \) and \( n + 30 \) for \( t = 3 \) and \( t = 4 \), respectively. These are significantly smaller than the complexity obtained in the previous section, whereas the worst case complexity is much larger.

Thus, for the (15,11) Hamming code the number of required additions is within the range 14–165 (14 when the sorting happens to be speediest and \( t = 0, 1 \)), but the neighborhoods of the two extremes are rarely reached. In particular, in the case of a channel with high signal-to-noise ratio, the complexity is confined, with high probability, to a narrow range centered at the vicinity of 42, because the probability that both \( i \) and \( j \) are large is rather small if the hard decisions \( b \) represent a single or double error. Notice that even when \( t \) is only 5, and, accordingly, the complexity is at most 47, it may occur that the decoded word is more likely than the one obtained by the Forney [6] or Chase [7] algorithms. (This may happen when \( i = 3 \) and \( j = 4 \), and also when \( j > 5 \).) In conclusion, for moderately noisy channels, this search scheme can be advantageous over that of Section II.

It is possible to reduce the complexity with the aid of a more elaborate scheme. For example, several other pairs may also be used as markers. Furthermore, summation over many quartets can be evaded by applying one or more triplets as markers, in addition to the pairs. Improvement is available also by incorporating into the scheme the method of Section II (see also Appendices I and II) for minimization over triplets and quartets that are derived from the same set of three and four pairs, respectively. Also note that the complexity of sorting, a significant portion of the total complexity for the case discussed, becomes negligible as \( n - k \) increases.

Both kinds of search schemes, of the previous and the present sections, are too complex for most codes, unless modified for suboptimal decoding. However, their application to the cosets of a subcode of some of those codes may yield optimal decoders with low complexity, as demonstrated in the next section.

IV. COSET DECODING

In most general terms, the approach taken hereafter for decoding a binary linear block code \( C \) consists of the following steps [1]:

a) find a subcode \( C_1 \), a basis for \( C_1 \) and a basis for the coset representatives;
b) given the received word, obtain the most likely
codeword in each coset, using a decoder for $C_1$;
c) compare the codewords obtained.

Efficiency considerations guide the otherwise arbitrary
selection of a subcode. Algorithms B and C of [2] are
specialized forms of this general decoding principle for $C_1$
of two conveniently decodable types, characterized by the
property that the set of the nonzero columns of a generator
matrix $G_1$ of $C_1$ has cardinality $\dim(C_1)$, resp. $\dim(C_1) + 1$.
A subspace of these types is utilized in all previously
published nontrivial implementations of the foregoing
coset-based decoding principle, although in [1] several
other candidates for $C_1$ were also suggested. Step b) for $C_1$
of the two types mentioned amounts to hard decision,
resp. single-check Wagner decoding, applied, for each coset
of $C_1$, to a vector each entry of which represents the soft
content of the symbols associated with a maximal set of
identical columns of $G_1$. The decoders of [1] and [2]
function this way, whereas the trellis decoders of [3]
operate in a related fashion. A similar utilization of other
subcodes is possible with the aid of Algorithm 1. This is
formally expressed by the following Algorithm 2.

Let the encoding of message word $s \in GF(2)^k$ be
described by $c = sg$, where $G$ is a generator matrix of $C$. Denote by $g_i; i = 0, 1, \cdots, n - 1$ the columns of $G$ and let
$\langle \cdot, \cdot \rangle$ denote inner product over $GF(2)$. According to (1),

$$M(s) = \sum_{i=0}^{n-1} (-1)^{\langle s, g_i \rangle} \mu(v_i),$$

where the modified notation in the left side emphasizes
that the sum is determined by $s \in GF(2)^k$. If $g_i = g_j$ for
$i \neq j$ (this occurs, for example, in a repetition code) then
replacement of $\mu(v_i)$ by $\mu(v_i) + \mu(v_j)$ in (8) allows deletion of
$g_j$. Therefore, we assume that $G$ has no repeated
columns.

Consider a subcode $C_1$ of $C$, generated by some matrix
$G_1$ with columns $\{g_i\}$. Assume that $C_1$ is contractible to a
$(J + \lambda, J)$ code for some $\lambda \geq 0$, which means, by definition,
that $\{\langle g_i; i = 0, 1, \cdots, n - 1 \rangle; \{0\} \} = J + \lambda$ where
$J = \dim(C_1)$. Choose for $C$ a generator matrix as follows:

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

where $G_2$ is a generator of the coset representatives. Now,
write $\{0, 1, \cdots, n - 1\} = \bigcup_{J+\lambda+1}^{J+\lambda+1} \Omega_j$, where $\Omega_j$ are location
sets such that $\{\langle g_i; j \in \Omega_j \rangle; \{0\} \} = J + \lambda$ and $\Omega_0$ is the (possibly empty) set which specifies
the location of the zero columns of $G_2$. Let $C_2$ be a so-called
contracted code generated by the matrix $D$ with columns
$\{d_i\}$ given by $d_0 = 0$ and $d_i = g_i$, where $j \in \Omega_0$, for all
$i = 1, 2, \cdots, J + \lambda$. Any decoder for $C_2$ is suited to be part of
a decoder for $C$. We shall apply Algorithm 1.

Assuming that $\lambda \geq 1$, let $H_D$ be a parity check matrix
of $C_D$. It is a $\lambda$ by $J + \lambda + 1$ matrix with an arbitrary 0th
column which is declared to be $0$. It follows by (8) that

$$M(s_1, s_2) = \sum_{i=0}^{J+\lambda} (-1)^{\langle s_1, d_i \rangle} \sum_{j \in \Omega_j} (-1)^{\langle s_2, g_j \rangle} \mu(v_j)$$

where we used the partitioning $s = (s_1, s_2)$ that corresponds to (9) and $\{g_j\}$ are the columns of $G_2$. Let

$$a_i(s_2) = \sum_{j \in \Omega_j} (-1)^{\langle s_2, g_j \rangle} \mu(v_j)$$

for $i = 0, 1, \cdots, J + \lambda$ and

$$b(s_2) = \left\{0, \text{sgn}[a_1(s_2)], \cdots, \text{sgn}[a_{J+\lambda}(s_2)]\right\} \in C_D.$$  

Obviously,

$$\left\{\langle s_1, d_0 \rangle, \langle s_1, d_1 \rangle, \cdots, \langle s_1, d_{J+\lambda} \rangle\right\} \in C_D;$$

hence by comparing (1) and (10) the following maximum
likelihood decoding rule is deduced.

**Algorithm 2**

a) Compute $a_i(s_2)$ given by (11) for all $s_2 \in GF(2)^{J+\lambda}$
and $i = 0, 1, \cdots, J + \lambda$.
b) For each $s_2 \in GF(2)^{J+\lambda}$ find an error location set
$L(s_2)$ by applying steps a)–c) of Algorithm 1 with
respect to the code $C_D$, with $b$ replaced by $b(s_2)$
given by (12).
c) Evaluate, for each $s_2 \in GF(2)^{J+\lambda}$,

$$M'(s_2) = a_0(s_2) + \sum_{i=1}^{J+\lambda} |a_i(s_2)| - 2 \sum_{i \in L(s_2)} |a_i(s_2)|.$$  

(14)
d) Maximize $M'(s_2)$. With an $s_2$, denoted $\hat{s}_2$, which
attains the maximum, find a solution $s_1$ to

$$c(\hat{s}_2) = s_1D$$

where $c(\hat{s}_2)$ is equal to $b(\hat{s}_2)$ complemented over
$L(\hat{s}_2)$. Decode as the message word the vector
$(s_1, \hat{s}_2)$.

$M'(s_2)$ is, of course, the largest value that $M(s_1, s_2)$
given by (10), attains by varying $s_1$ throughout $GF(2)^{J}$. For
the case $\lambda = 0$, $H_D$ is to be interpreted as “empty,” i.e.,
step b) is skipped and we set $L(\hat{s}_2) = \emptyset$.

For implementation of this algorithm tables [1], transforms
[2] and trellises [3] are available. Also, various search
schemes for step b) are applicable, and there are ways of
combining steps or performing them in parallel (examples
for the latter are [2, algorithm C and its iterative version]).
Of course, all these must be suited to the code and the
value of $\lambda$; the latter is at our disposal to some extent, as
demonstrated by Examples 4–6.

The computational complexity and memory requirement
are determined by the code and the subcodes formed by
Two possible patterns for the reduction of complexity, with an increase of $\lambda$, are exhibited by (16)-(20). One of them is the decrease of the values of $n_0, n_1, \ldots, n_{J+\lambda}$ as a consequence of increasing $J$. This occurs in all the examples presented later on. The other pattern is the decrease of the second term on the right side of the expressions. This is encountered, to some extent, in Example 5. The latter pattern usually requires a sufficiently steep increase or large initial value of $J$. Therefore, it is expected to participate more dominantly in the reduction of complexity for codes which are considerably larger than those we examined.

The zero column $d_0$ of $D$ was introduced with the case $\Omega_0 =\emptyset$ in mind, for associating the weight $a_0(x^2)$ with a location. Hence, whenever $\Omega_0 =\emptyset$, the leftmost column of both $D$ and $H_D$, together with the leftmost entry of $b(x^2)$, will be routinely punctured, and the coefficient of $x^{2^{J+\lambda}}$ in (16)-(20), is to be reduced by one. This condition is met throughout the following examples (see [2] for examples with $\Omega_0 =\emptyset$).

Example 4: In the (15,7,5) BCH code the largest value of $J$ for $\lambda = 0$, and apparently for $\lambda = 1$ as well, is three. Consider a subcode of this code generated by

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

where the first row specifies the generator polynomial. The subspace spanned by the last three rows of $E$ is geometrically similar to a (3,3) code. If its 16 cosets are decoded, then the complexity is $N_0 = 2 \cdot 2^3 + 3 \cdot 2^4 = 240$. Choosing, instead, the subspace spanned by $E$, which is contractible to a (6,4) code ($\lambda = 2$) but has only eight cosets, we obtain

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

for some version of $C_D$ and, consequently, as every 2-tuple but 0 is a column of $H_D$, $N_1 = (2 \cdot 2 + 2 \cdot 2^2 + 3 \cdot 2^3) + 11 = 132$. The complexity of Wolf's method [18] is 761.

Example 5: Let $C$ be the (17,9,5) BCH code, generated by $g(x) = 1 + x + x^2 + x^4 + x^6 + x^7 + x^8$. Consider first the following generator of a subcode of $C$:

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix},$$

for which $\lambda = 0$ and $N_0 = 384 + 3 \cdot 2^6 = 576$. Since $C$ has length 17 and minimum distance 5, the largest value of $J$ for $\lambda = 0$ is 3. Now let

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

for the partitioning (9), particularly by the values of $J$ and $\lambda$, as well as by the tool and procedures adopted. A small $\lambda$ tends to secure low complexity of step b) but may preclude a sufficiently large $J$ that would limit the complexity of the other steps as desired.

For any given $\lambda$, let $N_0, n_1, n_2, n_3, n_4, n_5$ be the complexity of Algorithm 2 in terms of real addition-equivalent operations. Let $n_i = |\Omega_i|$. Evaluation of $a_i(x^2)$ requires $(k - J)2^{k-1} - J$ additions with the aid of the fast Hadamard transform, but only $(n_i - 1)2^{n_i-1} - J$ additions under the assumption that $n_i < k - J$ and when, in compliance with [1] and [3], the complexity of determining absolute values is ignored. In many cases, depending on both $G_1$ and $G_2$, the complexity is further reducible with a more intensive use of tables, such as in [1]. We shall opt for that possibility in the next section. The complexity of step b) is determined by the average complexity and the number of applications of the search scheme (the latter can be less than $2^{k-1}$). Step c) requires $(J + \lambda)2^{k-1} - J$ additions when $n_0 > 0$. The complexity of step d) is $2^{k-1}$. Assume the following:

1. $n_i < k - J + 1$ for all $i = 0, 1, \ldots, J + \lambda$.
2. $n_0 > 0$.
3. the four steps of the algorithm are executed sequentially.
4. all nonzero $\lambda$-tuples are columns of $H_D$, and
5. the best search scheme described in Section II is applied $2^{k-1} - J$ times, and its worst case complexity is taken into account.

Then

$$N_0 = \sum_{i=0}^{J+1} (n_i - 1)2^{n_i-1} - (J + 1)2^{k-1} - J \quad (16)$$

$$N_1 = \sum_{i=0}^{J+1} (n_i - 1)2^{n_i-1} - (2J + 1)2^{k-1} - J \quad (17)$$

$$N_2 = \sum_{i=0}^{J+2} (n_i - 1)2^{n_i-1} - (2J + 2)2^{k-1} - J \quad (18)$$

$$N_3 = \sum_{i=0}^{J+3} (n_i - 1)2^{n_i-1} - (2J + 1)2^{k-1} - J \quad (19)$$

$$N_4 = \sum_{i=0}^{J+4} (n_i - 1)2^{n_i-1} - (2J + 7)2^{k-1} - J \quad (20)$$

For example, the coefficient $2J + 14$ in (19) is obtained by setting $\lambda = 3$ into $(J + \lambda + J + (J + \lambda) + 1 = 2J + 2\lambda + 8$, where the three terms on the left side correspond to steps b), c), and d), respectively. As all the assumptions, except for 1), represent worst case conditions, and a simple evaluation method for step a) was taken into account, complexities smaller than those expressed by (16)-(20) may be expected. Notice that (16) and (17) differ slightly from the corresponding expressions $N_0'$ and $N_1'$, respectively, of [2], because here the complexity of step d) is included in the formulas while the complexity of determining absolute values is ignored.
Then \( \lambda = 1 \) and a check matrix for a contracted code is

\[
H_D = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\
\end{bmatrix}.
\]

(23)

Because \( H_D \) has a zero column, in addition to the one that was punctured, the coefficient \( 2J + 2 \) in (17) is replaced by \( 2J \). Accordingly, \( N_4 = 106 + 8 \cdot 2^2 = 362 \). Allowing \( \lambda = 2 \), we may utilize the subspace \( C_4 \) generated by

\[
G_4 = \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{bmatrix}.
\]

(24)

\( C_4 \) is contractible to a (6, 4) code with check matrix

\[
H_D = \begin{bmatrix}
 0 & 1 & 1 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}.
\]

(25)

Hence \( N_4 = 36 + 13 \cdot 2^2 = 244 \). Notice that both terms of \( N_4 \) are smaller than the corresponding terms of \( N_2 \), although \( J \) increased by only one as \( \lambda \) changed from one to two. Furthermore, the reduction of the number of real addition-equivalent operations from 362 to 244 requires a negligible increase of logic. The complexity of Wolf’s method [18] is 2041.

Example 6: Consider the (18, 9, 6) code \( C' \) derived from the code \( C \) of Example 5 by extending the latter with an overall parity bit. The largest value of \( J \) for \( \lambda = 0 \) is three; it is attained, for example, by the extended version of \( G_4 \) given by (21). Accordingly, \( N_4 = 3 \cdot 5 \cdot 2^5 + 3 \cdot 2^5 = 672 \). By extending \( G_4 \), of (22), \( \lambda \) is raised to two. However, by augmenting \( G_4 \) of (21) with the codeword \((0, 3, 5, 12, 14)\) and subsequent extension, \( J = 4 \) is obtained with \( \lambda = 1 \). The corresponding complexity is \( N_4 = 192 + 8 \cdot 2^5 = 448 \). An all-ones column is appended to \( G_4 \) of (24). Hence a subcode of \( C' \), contractible to an (8, 5) code \( C_5 \), with check matrix

\[
H_D = \begin{bmatrix}
 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\end{bmatrix}.
\]

(25)

is obtained. By (19) \( N_5 = 36 + 23 \cdot 2^2 = 404 \). However, notice that \((1 \ 1 \ 0)\)' is not a column of \( H_D \), given by (25). Therefore, the complexity of decoding \( C_5 \) is 15 when the syndrome is \((1 \ 1 \ 0)\); otherwise, it is only ten for a nonzero syndrome. Hence, with 11 taken as average, the complexity of decoding totals \( 36 + 19 \cdot 2^2 = 340 \).

We did not examine all the sets of codewords of the codes considered in the foregoing examples, and it is quite likely that by other selections of a subcode decoders with still lower complexity are available. Nor did we attempt to exploit the properties of the cosets, in the fashion demonstrated in the next section for the (24, 12) Golay code. That approach is certainly rewarding for some other codes, and with respect to subcodes with various values of \( \lambda \).

V. Decoding the (24, 12) Golay Code

A generator matrix for the (24, 12, 8) extended Golay code \( C \) is given by

\[
G = \begin{bmatrix}
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}.
\]

(26)

Its first three and five rows span subspaces that are geometrically similar to (3, 3) and (6, 5) codes, respectively. In [1] these subspaces, and in [2] their punctured versions, were utilized for decoding. Also, \( G \) is identical, up to row-permutations, to a generator matrix upon which the differently operating trellis-decoder of [3] is based. Examination of \( G \), in an attempt to apply the multiple-check decoding method of the previous section, reveals that \( \lambda \geq 4 \) if \( J > 5 \) and, furthermore, \( J \leq 6 \) if \( \lambda = 4 \), for any subcode of \( C \) contractible to a \((J + \lambda, J)\) code. A subspace contractible to a \((10, 6)\) code does exist. For example, the first six rows of \( G \) span such subspace. The corresponding complexity is substantially less than what (20) predicts but exceeds the complexity of decoding the cosets of \( C \), the subspace generated by the first five rows of \( G \), as given by (17). According to the alternative approach, for increased efficiency, we shall decode the cosets of \( C \) while exploiting the structure of the first nine rows of \( G \). The exceptionally regular structure of the submatrix formed by these rows enables an explicit, yet simple, formulation of the decoding algorithm.

Let \( s_{ij} \) be the \( j \)th element of the \( i \)th column of \( G \). Partition the message word as follows:

\[
s = (s_1, s_2, s_3, s_4, s'),
\]

where \( s' = (s_5, s_6, \cdots, s_{11}) \). Let

\[
R_1 = (1 - 1) s_1 [(-1)^{s_6} r_0 + r_1] + [(-1)^{s_6} r_2 + r_3],
\]

\[
R_2 = (1 - 1) s_1 [(-1)^{s_6} r_4 + r_5] + [(-1)^{s_6} r_6 + r_7],
\]

\[
R_3 = (1 - 1) s_1 [(-1)^{s_6} r_8 + r_9] + [(-1)^{s_6} r_{10} + r_{11}],
\]

\[
R_4 = (1 - 1) s_1 [(-1)^{s_6} r_{12} + r_{13}] + [(-1)^{s_6} r_{14} + r_{15}],
\]

\[
R_5 = (1 - 1) s_1 [(-1)^{s_6} r_{16} + r_{17}] + [(-1)^{s_6} r_{18} + r_{19}],
\]

and

\[
R_6 = (1 - 1) s_1 [(-1)^{s_6} r_{20} + r_{21}] + [(-1)^{s_6} r_{22} + r_{23}],
\]

where

\[
r_i = (1 - 1) (e_{s_1} \cdot e_{s_2} \cdot e_{s_3} \cdot e_{s_4} \cdot e_{s_i}) \mu (e_i)
\]
for all \(i = 0, 1, \cdots, 23\). By (8), (26), and (27),

\[
M(s) = (-1)^{s_6}\left\{(-1)^{s_5}R_5 + R_6\right\} \\
+ (-1)^{s_5}\left\{(-1)^{s_4}R_4 + R_5\right\} \\
+ (-1)^{s_4}\left\{(-1)^{s_3}R_3 + R_4\right\}.
\]

Consequently,

\[
\begin{align*}
\max_s M(s) &= \max_s \left\{(-1)^{s_6}R_5 + R_6\right\} \\
&\quad + \left\{(-1)^{s_5}R_4 + R_5\right\} + \left\{(-1)^{s_3}R_3 + R_4\right\}.
\end{align*}
\]

(29)

Hence

\[
\begin{align*}
\max_s M(s) &= \left\{\begin{array}{ll}
\max_{s^6} \sum_{i=1}^6 |R_i|, & \text{if } \prod_{i=1}^6 R_i \geq 0 \\
\max_{s^6} \left\{\sum_{i=1}^6 |R_i| - 2|R_i|_{\min}\right\}, & \text{if } \prod_{i=1}^6 R_i < 0
\end{array}\right.
\end{align*}
\]

(30)

where \(|R|_{\min} = \min\{|R_i| : i = 1, 2, \cdots, 6\}\). Equation (30) is the basis of the most efficient decoding algorithms of [1] and [2] (compare it to [1, eq. (32)] and [2, eqs. (17)-(20)]).

Note that \(R_i\), for each \(i\), is assigned at most 16 different values as \(s_j\) vary, because it is expressed in terms of only four elements of \(\{\mu(t_j)\}\). Therefore, each \(|R_i|\) can have no more than eight different values.

It is convenient to rewrite (30) in terms of \(R_i^+\) and \(R_i^-\); \(i = 1, 3, 5\) given by

\[
R_i^+ = |R_i| + |R_{i+1}|
\]

and

\[
R_i^- = ||R_i| - |R_{i+1}||.
\]

(31)

Each of the three pairs \((R_i, R_{i+1})\) can be regarded as a function of only five variables, namely; \(s_2, s_4, s_6, s_10, s_{12}\) for \((R_1, R_2)\), \(s_1 \oplus s_3, s_4 \oplus s_5, s_9 \oplus s_{10}, s_{11}\) for \((R_3, R_4)\) and \(s_7, s_8, s_9, s_{10}, s_{11}\) for \((R_5, R_6)\). Therefore, each \(R_i^+\) and \(R_i^-\) assumes at most \(2^5 = 32\) different values. Now, for the case

\[
\prod_{i=1}^6 R_i \geq 0
\]

(32)

define

\[
M'(s^2) = R_1^+ + R_2^+ + R_3^+.
\]

(33a)

Otherwise, if \(\prod_{i=1}^6 R_i < 0\), let

\[
M'(s^2) = \begin{cases} 
R_1^+ + R_2^+ + R_3^+ & \text{if } |R|_{\min} = |R_1| \text{ or } |R_2| \\
R_1^- + R_2^- + R_3^- & \text{if } |R|_{\min} = |R_3| \text{ or } |R_4| \\
R_1^- + R_2^- + R_3^- & \text{if } |R|_{\min} = |R_5| \text{ or } |R_6|
\end{cases}
\]

(33b)

Evidently, \(\max_s M(s) = \max_{s^6} M'(s^2)\). This result and an examination of (29) and (28) imply the following maximum likelihood decoding rule (\(\text{sgn}(-)\) is defined by (3)).

**Algorithm 3:**

**Precomputation Stage:**

a) For each \(i = 1, 2, \cdots, 6\) compute the eight possible values of \(|R_i|\) and store the corresponding signs, \(\text{sgn}(R_i)\).

b) Compute the 32 possible values of each \(R_i^+\) and \(R_i^-; i = 1, 3, 5\).

**Main Stage:**

c) Compute \(M'(s^2)\), using (33), for the \(2^7 = 128\) possible values of \(s^2\) and find an \(s^2\), written \(\hat{s}^2\), which maximizes \(M'(s^2)\).

d) Decode as message word \(s = (s_0, s_1, s_2, s_3, s_4, s_5, s_6)\) where, assuming that (32) is satisfied for \(\hat{s}^2\),

\[
s_a = \text{sgn}(R_1) \oplus \text{sgn}(R_6)\]

\[
s_1 = \text{sgn}(R_1) \oplus \text{sgn}(R_2)\]

\[
s_2 = s_3 \oplus \text{sgn}(R_3)\]

\[
s_3 = s_4 \oplus \text{sgn}(R_4)\]

\[
s_4 = s_5 \oplus \text{sgn}(R_5)\]

\[
s_5 = s_6 \oplus \text{sgn}(R_6)
\]

(34)

and

\[
\text{sgn}(|R| - |R_{j+1}|) = 1.
\]

For performing step a), \(6 \times 10 = 60\) additions are sufficient, for each \(i = 1, 2, \cdots, 6\) a Gray-code ordering of the related elements of \(\{\mu(t_j)\}\) (see [1]). Step b) is performable by \(3(2^7 - 192) = 192\) additions. Thus the complexity of the precomputation stage totals \(60 + 192 = 252\). Note that the sign of \(|R| - |R_{j+1}|\) obtained during the computation of \(R_i^\pm\), specifies also \(R^\min = \min\{|R|, |R_{j+1}|\}\).

Computation of \(M'(s^2)\), for each \(s^2 \in \text{GF}(2)^7\), requires either two or four additions; the latter in case (32) is violated and, therefore, \(|R|_{\min} = \min\{|R|, |R_{j+1}|\}\) has to be found. Thus step c) is performed by at least \(2^8 \cdot 2 + 127 = 383\) and at most \(128 \cdot 4 + 127 = 639\) operations. However, there is an alternative more economical procedure.

For each of the 32 possible values of \((s_7, s_8, \cdots, s_{12})\) we maximize \(M'(s^2)\) with respect to \((s_7, s_8)\). Due to the structure of the generator matrix given by (26), \(R_1^+\) and \(R_1^-\) do not change as \((s_7, s_8)\) vary. Thus eight operations (five additions and three maximizations) are sufficient if condition (32) is fulfilled for all the four values of \((s_7, s_8)\).

Otherwise, two extra comparisons are required to obtain \(|R|_{\min} = \min\{|R|_{\min}, i = 1, 3, 5\}\) for each value of \((s_7, s_8)\) for which (32) is violated, and one extra addition is required in
case both $R_i^+$ and $R_i^-$ are needed for expressing the four values of $M(s^{2})$. Thus 17 additions are performed if (32) is violated for all the four values of $(s_{0}, s_{1})$. Finally, $\hat{s}$ is obtained by 31 additions. According to the latter scheme, the complexity of step c) is at least $32 \cdot 8 + 31 = 256$ and at most $32 \cdot 17 + 31 = 575$, or about 431 on average.

The number of addition-equivalent operations, required for the decoding, thus totals at most $252 + 575 = 827$, and it is about $252 + 431 = 683$ on average. Compare this to 1614 in [1], 1551 (1159 on average) in [2], and 1351 operations in [3]. (Rather than counting the numbers that appear in [1] and [2], we apply the method of assessing, as described earlier, to the algorithms of [1] and [2]).

The memory space occupied during step a) is 48 words, required for storing $R_i^+$, $R_i^-$, $R_i^l$, $R_i^m$, each with eight values. Regarding step b), $3 \cdot 2 \cdot 32 = 192$ memory words for $R_i^l$ and $R_i^m$, i = 1, 3, 5 and 3 \cdot 32 = 96 additional words for $R_i^{mn}$, i = 1, 3, 5 are required. As soon as step b) is completed, the information stored during step a) may be destroyed, except for $\text{sgn}(R_i^l)$ i = 1, 2, ..., 6. Thus 192 + 96 = 288 words are needed for the precomputation stage. Excluding a few memory words for temporary storage, no extra memory space is required for the main stage. Therefore, the total memory space is roughly 300 words.

However, a more careful look at steps b) and c) reveals that, due to the skewed structure of the middle four rows of $G$ in (26), not all the values of $R_i^+$, $R_i^-$, and $R_i^{mn}$ computed in step b), are simultaneously used according to the discussion of the computational complexity. Indeed, only four sets of $(R_i^+, R_i^-, R_i^{mn})$, four sets of $(R_i^+, R_i^-, R_i^{mn})$, and one set of $(R_i^+, R_i^-, R_i^{mn})$ have to be stored in memory at the same time. However, the 48 words from step a) must be preserved. Thus the total number of memory words, according to the latter scheme, is $48 + 3(4 + 4 + 1) = 75$. Compare this to the 6 \cdot 128 + 2 \cdot 16 + 4 \cdot 8 = 832 words required for the tables in [1], the memory space of 6 \cdot 128 = 768 words occupied by six vectors of Hadamard transform of order 7 in [2], and the 3 \cdot 64 + 8 = 200 words needed in [3] for storing the 64 metrics that belong to each of the three sections of the subtrellises with eight states each.

VI. DECODERS FOR THE (12, 6) GOLAY CODE

The general decoding principle, cited in Section IV, applies to block codes over any field as well as to lattice codes [1]. Also, the Wagner rule is extendable to other fields, but its implementation is usually quite intricate [16]. However, costest decoding of a ternary code with $\lambda = 0.1$ checks is simple. These are the cases that naturally arise when decoding of the (12, 6, 6) Golay code $C$ is considered, because $C$ may be generated by the following matrix:

$$
G = \begin{bmatrix}
111111 & 000000 \\
000000 & 111111 \\
000111 & 000111 \\
020001 & 010221 \\
010122 & 011211 \\
012201 & 012111
\end{bmatrix}.
$$

(35)

First a decoder based on the subspace geometrically similar to a (2, 2) code, spanned by the two top-most rows of $G$, will be presented.

Maximum likelihood decoding is now equivalent to seeking the source-word $s \in GF(3)^6$ that maximizes the expression:

$$
M(s) = \sum_{i=0}^{11} \log p(e_i | s, g_i) \tag{36}
$$

where $(\cdot, \cdot)$ stands for inner product over $GF(3)$. Let

$$
Q_i(s) = \max_{s_0} \sum_{i=0}^{11} \log p(e_i | s_0 + (s_i, g_i'))
$$

and

$$
Q_5(s) = \max_{s_5} \sum_{i=6}^{11} \log p(e_i | s_1 + (s_i, g_i')) \tag{37}
$$

where $s' = (s_0, s_1, s_2, s_3)$ and $g_i' = (g_{2i}, g_{2i+1}, g_{2i+2}, g_{2i+3}).$ Then by (35)

$$
\max_{s} M(s) = \max_s \left\{ Q_1(s') + Q_5(s') \right\} \tag{38}
$$

This leads to the following simple decoding procedure which is, in fact, a ternary counterpart of [2, Algorithm B] phrased to suit the particular code considered here.

Algorithm 4

Precomputation Stage:

a) Compute the $3^4 = 81$ possible values of $Q_i(s')$: $i = 1, 2, 4$, given by (37).

Main Stage:

b) Find a $s'$, call it $\hat{s}'$, for which (38) is satisfied.

c) For the source word $s$ set $s = (s_0, s_1, s')$ where $s_0$ and $s_1$ are values which maximize the expressions in (37) for $s' = \hat{s}'$.

It turns out to be more efficient to decode cosets of the subspace, geometrically similar to a (4, 3) code, which is spanned by the first three rows of $G$ given by (35). Let

$$
P_1 = \sum_{i=0}^{2} \log p(e_i | s_0 + (s^2, g_1'))
$$

$$
P_2 = \sum_{i=3}^{4} \log p(e_i | s_02 + (s^2, g_2'))
$$

$$
P_3 = \sum_{i=6}^{7} \log p(e_i | s_1 + (s^2, g_3'))
$$

$$
P_4 = \sum_{i=9}^{11} \log p(e_i | s_12 + (s^2, g_4'))
$$

where $s_02 = s_0 + s_2$, $s_12 = s_1 + s_2$, $s^2 = (s_0, s_2, s_3)$ and $g_i' = (g_{2i}, g_{2i+1}, g_{2i+2}, g_{2i+3}).$ Notice that each $P_i$: $i = 1, 2, 3, 4$ may have at most $3^3 = 27$ distinct values. Furthermore, for a fixed $s^2$, each $P_i$ is assigned at most three different values as $s_0$, $s_1$, and $s_2$ vary. Therefore, the 27 values of $P_i$ can be grouped into nine sets of three, where each set corresponds to some subset of $\{s^2\}$. It proves advantageous to sort the three
elements of each such set. Denote the values by $P_{i}^{+}$, $P_{i}^{-}$,
and $P_{i}^{\pm}$; $i = 1, 2, 3, 4$, where $P_{i}^{+} \geq P_{i}^{\pm} \geq P_{i}^{-}$. Let $s_{a(i)}^{i}, s_{s(a)}^{i}, s_{s(a)}^{+}$
and $s_{a(i)}^{-}$ be, respectively, the arguments for which these values
(with some fixed $s_{i}^{2}$) are attained, where $x(\cdot)$ is defined by
$x(1) = 0$, $x(2) = 0$, $x(3) = 1$, and $x(4) = 12$. For instance,
$P_{i}^{+}(s_{a(i)}^{2}) = P_{i}^{+}(s_{s(a)}^{i}, s_{s(a)}^{2}) = \max_{s_{a(i)}^{i}} P_{i}(s_{a(i)}^{i}, s_{s(a)}^{2})$, and
$P_{i}^{-}(s_{a(i)}^{2}) = P_{i}^{-}(s_{s(a)}^{i}, s_{s(a)}^{2}) = \min_{s_{a(i)}^{i}} P_{i}(s_{a(i)}^{i}, s_{s(a)}^{2})$.

Clearly, if the following "syndrome"
\[ \rho = (s_{a(i)}^{i} - s_{a(i)}^{-}) - (s_{a(i)}^{i} - s_{a(i)}^{-}) \] (39)
satisfies \[ \rho = 0 \pmod{3} \], then
\[ \max_{s_{i}^{2}} M(s) = \max_{s_{i}^{2}} \sum_{i=1}^{4} P_{i}^{\pm}(s_{i}^{2}). \] (40)

Otherwise, if $\rho = 1$ or $2$ (modulo 3), then for one or more
$i \in \{1, 2, 3, 4\}$, the $P_{i}^{\pm}$ in (40) have to be replaced by $P_{i}^{\pm}$
or $P_{i}^{-}$ in a way that the right side of (39), with superscripts
appropriately changed, becomes 0 (modulo 3), while $M(s)$ is maintained as large as possible. We shall formulatce this explicitly. Let
\[ \Delta_{i}^{+} = P_{i}^{+} - P_{i}^{+} \]
\[ \Delta_{i}^{-} = P_{i}^{-} - P_{i}^{-} \] (41)
and set $\Delta_{i}^{0} = 0$. Let
\[ \Delta_{i}^{d} = \begin{cases} \Delta_{i}^{+}, & \text{if } s_{a(i)}^{i} - s_{a(i)}^{-} = d \pmod{3} \\ \Delta_{i}^{-}, & \text{if } s_{a(i)}^{i} - s_{a(i)}^{-} = d \pmod{3} \end{cases} \]
for $i = 1, 2, 3, 4$ and $d = 1, 2$. Notice that all $\Delta_{i}^{d}$ are obtained,
and the relation $\Delta_{i}^{d} \geq \Delta_{i}^{0}$ for each $i = 1, 2, 3, 4$ may
also be established, during the ordering process of the sets
$\{P_{i}^{+}, P_{i}^{\pm}, P_{i}^{-}\}$ with no additional computational cost. Let
\[ \nabla_{i}^{d} = \min \{ \Delta_{i}^{d}, \Delta_{i+1}^{d}, \Delta_{i}^{d} + \Delta_{i+1}^{d} \} \]
and
\[ \nabla_{i}^{0} = \min \{ \Delta_{i}^{0}, \Delta_{i+1}^{0}, \Delta_{i}^{0} + \Delta_{i+1}^{0} \} \]
for $i = 1, 2, 3$. (\nabla_{i}^{d} is the minimum possible decrease of $\Sigma_{i=1}^{4} \rho_{i}$
caused by exchanging the superscript of one or both of $P_{i}^{+}$
and $P_{i}^{-}$, so that the induced change on $s_{a(i)}^{i} - s_{a(i)}^{i}$
the first term on the right side of (39), is a reduction by $d$ (modulo 3).) It follows that
\[ \max_{s_{i}^{2}} M(s) = \max_{s_{i}^{2}} \left\{ \sum_{i=1}^{4} P_{i}^{\pm}(s_{i}^{2}) - \delta_{\rho}(s_{i}^{2}) \right\} \] (42)
where $\delta_{\rho}$ is given by $\delta_{\rho} = 0$,
\[ \delta_{1} = \min \{ \nabla_{1}^{1}, \nabla_{3}^{1}, \nabla_{2}^{1} + \nabla_{3}^{1} \} \]
and
\[ \delta_{2} = \min \{ \nabla_{2}^{2}, \nabla_{3}^{2}, \nabla_{2}^{2} + \nabla_{3}^{2} \}. \] (43)

An alternative representation of (42) is obtained by observing that
\[ \delta_{\rho}(s_{i}^{2}) = \sum_{i=1}^{4} \Delta_{i}^{\rho(i)}(s_{i}^{2}) \] (44)
with appropriately selected superscripts $\rho(i)$; $i = 1, 2, 3, 4$
where $\rho(i) \in \{+, -, -\}$ (thus $\Delta_{i}^{\rho(i)}$ is defined by (41)).
We remark that, for a given $s_{i}^{2}$, $\delta_{\rho}$ assumes only 31 values
(1 for $\rho = 0$ and 15 for each $\rho = 1, 2$), even though there
appear to be $3^{4} = 81$ possible values. Now let
\[ M'(s_{i}^{2}) = \sum_{i=1}^{4} P_{i}^{\rho(i)}(s_{i}^{2}) \] (45)
where $\rho(i)$ for all $i = 1, 2, 3, 4$ are determined in accordance
with (44) and the values of $\rho$ and $s_{i}^{2}$. Evidently,
\[ \max_{s_{i}^{2}} M(s_{i}^{2}) = \max_{s_{i}^{2}} M'(s_{i}^{2}), \]
thus proving the following maximum likelihood decoding rule.

\subsection{Algorithm 5:}

\textbf{Precomputation stage:}

\textbf{a)} For each $i = 1, 2, 3, 4$ compute the 27 possible values
of $P_{i}^{+}$, and arrange them in nine ordered sets
$\{P_{i}^{+}, P_{i}^{\pm}, P_{i}^{-}\}$. Also store the corresponding values
of $\Delta_{i}^{+}, \Delta_{i}^{-}$ and $s_{a(i)}^{i}, s_{s(a)}^{i}$, (\nabla_{i}^{d}(i))

\textbf{Main stage:}

\textbf{b)} For each of the $3^{2} = 27$ possible values of $s_{i}^{2}$ compute
$M'(s_{i}^{2})$, using (45), and find an $s_{i}^{2}$, say $s_{i}^{2}$, which
maximizes $M'(s_{i}^{2})$.

c)} Decode as the source-word $s = (s_{0}, s_{1}, s_{2}, s_{3})$, where
\[ s_{0} = s_{a(1)}^{0} \]
\[ s_{1} = s_{a(3)}^{1} \]
\[ s_{2} = s_{a(2)}^{2} - s_{a(1)}^{0} \pmod{3} \]
with $\rho(i)$ determined in accord with (44) and the
values of $\rho$ and $s_{i}^{2}$, and all $s_{a(i)}^{i}$ are evaluated for $s_{i}^{2}$.
Along the lines of the discussion of complexity in
Section V it can be easily verified, following steps
a)–c) of Algorithm 5, that this decoder requires
$4[(9 + 27) + 3 \cdot 9] = 252$ additions for the precomputation stage and at most $2724 + 3 + 3 + 26 = 404$ operations for the main stage, a total of at most 656 additions. The complexity of the trellis decoder of
[4] is 1061 additions. The simpler decoder of Algorithm 4 requires $2[(9 + 27) + 243 + 2 \cdot 81] = 954$ additions for the precomputation stage and $81 + 80 = 161$ operations for the main stage, yielding a total of
1115 additions. The memory complexity of both
Algorithms 4 and 5 is less than 100 words.

\textbf{Acknowledgment}

The authors wish to thank G. D. Forney, Jr., who kindly
provided preprints of his papers, and are indebted to the
referees for their useful comments that helped improve the
presentation of this paper.

\textbf{Appendix I}

An implementation of Algorithm 1 is described here for the
case $n = k = 4$, with every 4-tuple except $0$ being a column of $H$. 
Consider step c), and let the 15 vectors involved be represented by their binary value. Due to the symmetry we may assume, without loss of generality, that (unless \( z = 0 \)) the syndrome \( z \) is all ones, i.e., 15. Then the pairs are (1,14), (2,13), (3,12), (4,11), (5,10), (6,9), and (7,8). Most of the time these are considered to be unordered pairs; parentheses are used for convenience. The sets for deriving triplets are the following:

\[
\begin{align*}
\{ (1,14), (2,13), (3,12) \} \\
\{ (1,14), (4,11), (5,10) \} \\
\{ (1,14), (6,9), (7,8) \} \\
\{ (2,13), (4,11), (6,9) \} \\
\{ (2,13), (5,10), (7,8) \} \\
\{ (3,12), (4,11), (7,8) \} \\
\{ (3,12), (5,10), (6,9) \}
\end{align*}
\]

whereas quartets are formed with the aid of the sets

\[
\begin{align*}
\{ (1,14), (2,13), (4,11), (7,8) \} \\
\{ (1,14), (2,13), (5,10), (6,9) \} \\
\{ (1,14), (3,12), (4,11), (6,9) \} \\
\{ (1,14), (3,12), (5,10), (7,8) \} \\
\{ (2,13), (3,12), (4,11), (5,10) \} \\
\{ (2,13), (3,12), (6,9), (7,8) \} \\
\{ (4,11), (5,10), (6,9), (7,8) \}
\end{align*}
\]

For example, both triplets (1,2,12) and (14,2,3) are obtained from the first set listed in (46). One of the quartets corresponding to the last set of (47) is (4,5,6,8). The rest of the quartets corresponding to the same set are obtainable by replacing elements two at a time, by their complements. They are (4,5,9,7), (4,10,6,7), (4,10,9,8), (11,10,6,8), (11,10,9,7), (11,5,6,7), and (11,5,9,8).

To apply the more efficient method of minimization, as described in Section II for the case \( n - k = 3 \) and in Example 2, first locate the column associated with the less reliable bit in each of the pairs. This requires seven additions and, as a byproduct, also yields the absolute value of the difference between the confidence values, called increment for short, over each pair. If sets of three (resp. four) columns thus obtained form a triple (resp. a quartet) according to each of the sets of pairs in (46) and (47), then the summation of confidence values over the best seven candidates of triplets and seven candidates of quartets is performed by altogether \( 7 \times 2 - 7 \times 3 = 42 \) additions. In the worst case, minimization of the increments over each of the sets of three and four pairs in (46) resp. (47) is also needed. This requires \( 21 \) additions (7-2 = 14 for (46) and seven more for (47)). In any case, the final minimization is over 22 numbers. Thus the total complexity is at least \( (n - 15) \times 7 + 1 \times 2 + 42 + 21 = n + 55 \) and at most \( n + 55 + 21 = n + 76 \) real additions. Instead of minimizing the increments over the various sets of pairs, one may prefer to arrange them in nondecreasing order. This requires no more than 13 additions [19]. Accordingly, the complexity is at most \( n + 68 \).

We remark that the sets of (46) are the three sets of a 2-(7,3,1) design, whose seven points are the seven pairs. Furthermore, the ovals [20] of this design are the sets listed in (47); they form a 2-(7,4,2) design. This is presumably extendible to some other cases, where in step c) we encounter a Hamming or a related code (such as a Reed-Muller code), thereby facilitating the description and perhaps the implementation of decoders based on Algorithm I.

**APPENDIX II**

The complexity of the search scheme of Section III is considered here for the case \( n = k = 4 \), when the most unfavorable situation \( i = 7, j = 8, \) and \( t = 15 \) prevails. To enumerate the triplets and quartets over which the confidence values have to be summed, it is possible to resort, without loss of generality, to (46) and (47). This follows by observing that (7,8) is a pair by assumption and, furthermore, any other pair has one location label smaller than seven and another which exceeds eight. Of course, the (original) location labels of two quartets derived from distinct sets in (46) do not necessarily add up, when expanded in radix-2, to the same vector. A similar comment applies to quartets. Consider a set of (46) that does not contain the pair (7,8). If one of the triplets generated by it has no label that exceeds six, then according to the permutation rule, each of the three other triplets have two labels that exceed eight. On the other hand, if one triplet has no label smaller than nine, then each of the three other triplets have precisely one label that exceeds eight. Thus either one or three triplets generated by such a set in (46) are checked by the search scheme; whereas a set that contains (7,8) generates one triplet that is checked. Hence, at least seven and at most 15 triplets are checked. Straightforward minimization over 15 triplets requires \( 15 \times 2 - 15 - 44 \) additions. A set in (47) that contains the pair (7,8) generates only one quartet that has to be considered. Any other set in (47) generates either one or four such quartets subject, however, to the constraint that at least one of the sets generates four quartets that are checked. Thus at least 11 and at most 16 quartets are checked. Straightforward minimization over 16 quartets requires \( 16 \times 3 + 15 = 63 \) additions. Consequently, the worst case complexity totals \( (n - 15) + 42 + 7 + 44 + 63 + 9 = n + 150 \). However, the complexity corresponding to only seven triplets and 11 quartets, scored straightforwardly, is only \( (n - 15) + 42 + 7 + 20 + 43 + 9 = n + 106 \) even with the maximum complexity 42 of sorting taken into account.

**REFERENCES**


[10] ______, “The construction of fast, high-rate, soft decision block


