A New Decoding Algorithm for Geometrically Uniform Trellis Codes

Sergio Benedetto, Senior Member, IEEE, and Guido Montorsi, Member, IEEE

Abstract—A new decoding algorithm for geometrically uniform trellis codes is presented. The group structure of the codes is exploited in order to improve the decoding process. Analytical bounds to the algorithm performance and to its computational complexity are derived. The algorithm complexity does not depend on the number of states of the trellis describing the code. Extensive simulations yield results on the algorithm performance and complexity, and permit a comparison with the Viterbi algorithm and the sequential Fano algorithm.

Index Terms—Geometrically uniform codes, group codes, sub-optimal decoding algorithms.

I. INTRODUCTION

RECENTLY, a paper by Forney [1] has illuminated some group properties of trellis coded modulation (TCM) schemes which form the basis for a new point of view on the codes structure, their construction and their performance evaluation. The main result in [1] was the definition of a new class of TCM codes, called geometrically uniform (GU) codes, which exhibit nice symmetry properties in terms of Voronoi regions. In fact, for GU TCM codes, the Voronoi regions are congruent, which essentially means that the characteristics of the codes in terms of Euclidean distance and error probability can be estimated by assuming that the encoded all-zero sequence has been transmitted. This property, often called uniform error property, dramatically simplifies the performance analysis. Although the geometrical uniformity greatly simplifies both the search for good codes and the error performance analysis, there has been no direct impact on the maximum likelihood (ML) decoding mechanism, which is still based on the Viterbi algorithm. Thus, in the conclusion of its paper, Forney presented, among other topics for future research, the following question "... can the symmetry of more general geometrically uniform codes be exploited in decoding?"

In this paper, we try to move some steps toward an answer to Forney’s question. Exploiting the group properties of GU codes, we devise a new decoding algorithm whose complexity is independent of the number of states of the code trellis. Just like the Viterbi algorithm, it can be stated in a version which performs the ML detection of the received sequence and implemented in suboptimum versions to cope with practical requirements.

The algorithm will be described with reference to GU TCM codes, however, it can obviously be applied as well to convolutional codes over groups. In Section II, we recall some properties of GU trellis codes, which are necessary to understand the proposed algorithm, and in Section III we rephrase the principles of ML sequence detection using the notations applied in the following to describe the new algorithm. Section IV is devoted to the description of the new algorithm and to its adaptation to the case of GU TCM based on $L \times M$-ary phase shift keying (MPSK) signal constellations. The computational complexity of the algorithm is discussed in Section V, both in terms of simulated and analytical results. In Section VI, we develop analytical upper bounds to the performance of the algorithm. In Section VII, the new algorithm is compared with the Viterbi and Fano sequential algorithms in terms of performance and complexity. Finally, in Section VIII, we propose a further simplification of the algorithm which significantly reduces its complexity.

II. GEOMETRICALLY UNIFORM TCM CODES

In this section, we briefly recall the definitions and results of [1] and [2] which will be used in the rest of the paper. The reader is referred to these papers for proofs and deeper insights into the matter.

We consider signal constellations in the real $N$-dimensional space $R^N$ with the associated Euclidean distance metric $d^2(x, y) = |x - y|^2$, for $x, y \in R^N$. An isometry is a mapping $f : R^N \to R^N$ such that, given any two points $x, y \in R^N$, $d^2[f(x), f(y)] = d^2(x, y)$. An isometry mapping a constellation $X$ to itself is called a symmetry, and the sets of symmetries form a group under composition of mappings, called the symmetry group $\Gamma(X)$. As an example, if we denote as $R_M$ the group of rotations by multiples of $360/|M|$ degrees and by $V$ the group consisting of the identity and the reflection about the line between any point or midpoint of the constellation and the origin, the symmetry group of the MPSK signal constellation consists of all compositions of elements of $R_M \bigcirc V$, which we denote by $R_M \bigcirc V$.

A given signal set $X$ is said to be GU if, for any two points $x$ and $x'$ in $X$, there exists a symmetry of $X$ that sends $x$ into $x'$. The generating group $G(X)$ of $X$ is a subgroup of $\Gamma(X)$ which is minimally sufficient to generate $X$ starting from an
For codes presenting parallel transitions in the trellis, i.e., with more than one edge joining the same pair of states, the ML decoding can be performed in two steps: the first step chooses the individual signal within the subset \(a(k)\) of signals associated with the parallel transitions (which together form a branch of the trellis) as the one closest to the received signal, store it and associate with the corresponding branch a branch metric equal to its distance from the received signal, namely

\[
m[a(k)] = \min_j \{d^2[a_i(k), y(k)]\}.
\]

The second step performs an ML sequence detection on the reduced trellis obtained during the previous step based on the path metrics

\[
M(A) = \sum_{k=\infty}^{\infty} m[a(k)].
\]

For a general TCM code with \(N_a\) states, the minimum complexity algorithm performing the ML decoding is the Viterbi algorithm. It is a recursive algorithm minimizing the path metric (2) step-by-step on a truncated code defined on the time interval \([-\infty, k]\). The set of sequences to be considered in the minimization at step \(k + 1\) is the set of all possible extensions of the sequences with smallest metrics reaching the \(N_a\) states at time \(k\) (the surviving paths). The path metrics \(M(S_{k+1})\) of the sequences \(S_{k+1}\) defined in the interval \([-\infty, k + 1]\) are thus computed using those at step \(k\) and the branch metrics at step \(k + 1\) through

\[
M(S_{k+1}) = \sum_{j=-\infty}^{k+1} m[s_{k+1}(j)]
\]

where \(S_{k+1} = [S_k, s_{k+1}(k + 1)]\).

To be a truly ML algorithm, the Viterbi algorithm should wait until the end of transmission before taking a decision on the transmitted sequence. In practice, this is not possible, so that a decision on the first signal of the transmitted sequence has to be made after a suitably chosen decoding delay.

In the following, we explain the new algorithm in its basic version, then introduce a slight modification and specialize it to phase shift keying (PSK)-like signal constellations. Finally, we derive a suboptimal version of the algorithm, which permits a sensible reduction in the complexity with a slight performance degradation.

IV. DESCRIPTION OF THE NEW ALGORITHM

The proposed algorithm exploits the group structure of the states and sequences of any length in the code in order to retrieve, at each step \(k\), the ML sequence \(S^{(ML)}_k\) starting from the knowledge of \(S^{(ML)}_{k-1}\).

Let \(S_{k-1}^{(ML)}\) be the ML sequence at step \(k - 1\) and \(S_k^{(ML)}\) its final state. Instead of extending all the survived sequences (one for each state) to find the ML sequence at step \(k\), the algorithm considers the sequence obtained extending \(S_{k-1}^{(ML)}\) with the label \(s_k^{(P)}(k)\) to which the minimum branch metric
is associated

\[ S_k^{(P)} = [S_{k-1}^{(ML)}, s_k^{(P)}(k)]. \]

This sequence represents a reasonable first candidate to the ML sequence at step \( k \) in the sense that \( D[S_k^{(P)}, s_{k-1}^{(ML)}] \) is probably small.

We have at this point two possibilities. Either \( S_k^{(P)} \) is the ML sequence at step \( k \), or the ML sequence at step \( k \) is a sequence that diverged from \( S_k^{(P)} \) at some step \( j \) in the past and never merged into it at a time before \( k \). In fact, all sequences diverging from \( S_k^{(P)} \) and merging into it at a time before \( k \) cannot be part of the ML sequence \( s_{k-1}^{(ML)} \) which is included in \( S_k^{(P)} \) (see Fig. 1). Furthermore, if the ML sequence at step \( k \) is not \( S_k^{(P)} \), then its branch metric relative to the last step must be smaller than the branch metric relative to \( s_k^{(P)}(k) \).

Owing to the group structure of the code, the ML sequence at time \( k \) can be obtained by adding to \( S_k^{(P)} \) a suitable error sequence \( S_e \) and to the state \( \sigma_k^{(P)} \) the state \( \sigma_e \). The additions must be performed using the proper group operations, namely

\[ S_k^{(ML)} = S_k^{(P)} \oplus S_e, \]

\[ \sigma_k^{(ML)} = \sigma_k^{(P)} \oplus \sigma_e. \]

To find the ML sequence \( S_k^{(ML)} \), we must try all possible error sequences, compute the path metrics of the new candidate sequences obtained through the sequence group addition, and compare those metrics with \( M[S_k^{(P)}] \). Once the final ML sequence has been obtained, its starting state and symbol uniquely determine the decoded bits. Owing to the group properties of the code, the set of error sequences is independent of the considered sequence and is equal to the set of sequences diverging from the identity sequence \( 0 \). Thus, the table of error sequences can be obtained by adding to \( S_k^{(P)} \) a suitable error sequence \( S_e \) and to the state \( \sigma_k^{(P)} \) the state \( \sigma_e \). The additions must be performed using the proper group operations, namely

\[ S_k^{(ML)} = S_k^{(P)} \oplus S_e, \]

\[ \sigma_k^{(ML)} = \sigma_k^{(P)} \oplus \sigma_e. \]

A crucial point of the algorithm is a set of rules that allows to drop out candidate sequences. The geometrical properties of GU codes help once more in limiting the number of sequences to be examined.

Let \( A = (a_{k-h}, \ldots, a_k) \), \( B = (b_{k-h}, \ldots, b_k) \), and \( R = (r_{k-h}, \ldots, r_k) \) be the last \( h + 1 \) elements of the extension sequence, a candidate sequence and the received sequence. As a result of the addition of an error sequence of length \( h + 1 \), the candidate sequence differs from \( A \) only by the last \( h + 1 \) labels, and consequently the comparison can be performed considering only these last labels. \( A, B, \) and \( R \) can be viewed as three points in a Euclidean space of dimension \( 2L(h + 1) \) where \( 2L \) is the dimensionality of the signal space (see Fig. 2).

Consider now, the distance

\[ D^2(B, R) \triangleq |B - R|^2 = |B - A + A - R|^2 \]

\[ = |B - A|^2 + |A - R|^2 + 2(B - A)(A - R) \]

\[ = D^2(A, R) + D^2(A, B) + 2(B - A) \cdot (A - R) \]

where "·" denotes the scalar product.

The following relation is easily obtained

\[ D^2(B, R) \geq D^2(A, R) \]

\[ \Rightarrow D^2(A, B) + 2(B - A) \cdot (A - R) \]

\[ \geq 0. \]
Minimizing the expression for the scalar product
\[ D^2(A, B) - 2|B - A||A - R| \geq 0 \]
\[ \rightarrow D^2(A, B) \geq 2D(A, B)D(A, R) \]
we can conclude that the candidate sequence \( B \) can be neglected if
\[ D^2(A, B) \geq 4D^2(A, R). \]

Owing to the group properties of GU codes and since \( B = A + S_e \), from (1) we get
\[ D(A, A + S_e) = D(S_e, 0) \]
\[ \triangleq D_e \]
so that the comparisons can be limited to the error sequences satisfying
\[ D_e \leq 2D(A, R) \]
\[ \triangleq T_1. \]

In the following, we will call \( D(A, R) \) the \((h + 1)\)-truncated metric of the extension sequence as it is obtained summing up the last \( h + 1 \) branch metrics of the extension sequence.

The \((h + 1)\)-truncated metric is a random variable (rv) assuming values in the range \((0, \infty)\) and, thus, especially for low signal-to-noise ratios (SNR’s), the number of error sequences to be examined can become unlimited.

To overcome this problem in an actual implementation of the algorithm, we can fix a threshold \( d_{\text{max}} \) to the value \( D_e \), which in turn defines a value \( h_{\text{max}} \) as the length of the longest sequence \( S_e \) with distance \( D_e \leq d_{\text{max}} \).

This poses an upper bound to the storage requirements, but leads to a suboptimal version of the algorithm, in the sense that it presents a failure probability, i.e., the probability of losing the ML path. The failure probability can be controlled by choosing the value \( d_{\text{max}} \) according to an estimate of the statistics of the \( h \)-truncated metrics obtained by simulation with the foreseen value of the SNR.

An important feature of the proposed algorithm is that the number of comparisons reduces when the SNR increases, as the \((h+1)\)-truncated metrics of the extension sequence become smaller.

### A. Improving the Algorithm

We have seen in the previous subsection that the ML sequence up to step \( k \) should end only with a branch whose metric is smaller than the metric corresponding to the metric of the extension sequence. This property was not exploited in the previous algorithm and will be considered here to reduce its complexity.

Define \( F_1 \) as the set of labels stemming from the identity state \( \sigma_1 \). In [2], it was proved that \( F_1 \) is actually a normal subgroup (the input group) of the set \( E \) of all labels used in the trellis. Therefore, we can define the quotient group
\[ \frac{E}{F_1} = \{F_1, \cdots, F_p\} \]
\[ p = \frac{|E|}{|F_1|}. \]

The mapping \( \varphi : \Sigma \rightarrow E/F_1 \), assigning to each state the set of output labels stemming from it, is a homomorphism, defining a partition of the state group
\[ \Sigma \]
\[ \Sigma_1 = \{\Sigma_1, \cdots, \Sigma_p\} \]
where \( \Sigma_1 \triangleq \text{Ker}(\varphi) \) is the set of states whose output labels belong to \( F_1 \). The states belonging to the same class \( \Sigma_i \) share the same set of transition labels \( F_i \) (Fig. 3).

The next step consists of constructing \( p \) tables \( M_i \) of error sequences, one for each element of the partition \( \Sigma/\Sigma_1 \). Each table contains the error sequences \( S_e' \) ending in all states \( \sigma_e' \) such that \( \varphi(\sigma_e') = F_i \).

Beyond the error sequence, each table \( M_i \) contains the terminal state \( \sigma_e' \) and the Euclidean distance of the error sequence from the identity sequence. The sequences are sorted in an ascending way on the length and on the Euclidean distance. The algorithm, at each step, proceeds as follows.

Together with the branch metric computation it identifies the best metric class labels
\[ f_{i}^{\text{min}} \triangleq \arg\min_{f_{j} \in F_i} m(f_{j}) \quad \forall i = 1, \cdots, p. \]

Only one of these labels can be the ending label of the ML sequence. Furthermore, the extension sequence is formed by
concatenating the ML sequence at step \( k - 1 \) with \( f_{k-1}^{\text{min}} \) such that \( F_k = \varphi[\sigma_{k-1}^{(M,L)}] \).

The state at step \( k - 1 \) of a possible candidate sequence must belong to one of those \( \Sigma_i \) such that \( m(f_{k-1}^{\text{min}}) < m(f_{k}^{\text{min}}) \). If, for a given \( M_i \) [with \( \sigma_{k}^{(M,L)} \) any state such that \( \varphi(\sigma_{k}^{(M,L)}) = F_k \)]
\[
\varphi(\sigma_{k-1}^{(M,L)} + \sigma_{k}^{(M,L)}) = F_k
\]
and
\[
m(f_{k-1}^{\text{min}}) \geq m(f_{k}^{\text{min}})
\]
we can skip all the entries in the table. When the branch metric of the extension sequence \( m(f_{k}^{\text{min}}) \) is the best branch metric, this criterion drops all the possible tables, thus reducing considerably the complexity of the algorithm.

Beyond this saving, the new approach slightly reduces the threshold (3). In fact, the set of error sequences \( S_e \) considered now are the concatenation of the error sequences \( S'_e \) belonging to \( M_i \) and the best last branch \( f_{k-1}^{\text{min}} \)
\[
S_e = (S'_e, f_{k-1}^{\text{min}}).
\]
The threshold must be set with respect to the Euclidean distances \( D_e' \) stored in \( M_i \), corresponding to error sequences one step shorter than those considered before.

Using the same notation as in the previous section, we can break down \( A, B, \) and \( R \) as follows:
\[
A = (A', a_k)
\]
\[
B = (B', b_k)
\]
\[
R = (R', r_k).
\]

Then, following the same procedure that led to (3), we obtain the new threshold
\[
D'_e \leq D(A', R') + \sqrt{D^2(A', R') + \Delta_1} \equiv T'_1
\]
where
\[
\Delta_1 \equiv D^2(a_k, r_k) - D^2(b_k, r_k).
\]
The threshold \( T'_1 \) can be easily proven to be always less than (or equal to) the threshold \( T_1 \) previously defined in (3).

To make better understandable the proposed algorithm, we have represented its main steps in the flow chart of Fig. 4. The flow chart includes also the refinement to the algorithm which will be discussed in Section VIII.

### B. Matching the Algorithm to Multidimensional L×MPSK GU Codes

An interesting property of group codes designed on multidimensional PSK constellation is that all sequences lie on a hypersphere of radius \( \infty \). As a consequence, instead of considering Euclidean distances, we can decide on the basis of scalar products.

Thus, for ML decoding purposes, a received point \( R \) is equivalent to any point of the radius of the hypersphere passing through it. In order to set the threshold, it is then convenient to take as received point the point lying on this radius closest to the sequence \( A \) (Fig. 5).

For \( L \times \text{MPSK} \) constellations the inequality (3) becomes
\[
D_e \leq 2 \sqrt{hL - \frac{A' \cdot R'}{|R'|^2} (A' \cdot R' - \Delta_2)} \quad \Delta \equiv T_2
\]

Taking the difference between \( T_1^2 \) and \( T_2^2 \)
\[
T_1^2 - T_2^2 = 4\left(1 - \frac{A \cdot R}{R^2}\right)^2
\]
we see that it is always greater or equal than zero (\( T_1 = T_2 \) when \( A \cdot R = R^2 \)).

Proceeding as for the general case, we can obtain the threshold corresponding to \( T'_1 \)
\[
D'_e \leq 2 \sqrt{hL - \frac{A' \cdot R'}{|R'|^2} (A' \cdot R' - \Delta_2)} \quad \Delta \equiv T'_2
\]

where
\[
\Delta_2 = b_k \cdot r_k - a_k \cdot r_k.
\]

Threshold (8) is the one implemented in the algorithm and used in the examples and results that follow.

### V. Computational Complexity

In this section, we compare the computational complexity of the proposed algorithm with the Viterbi algorithm and with the Fano sequential algorithm.

The algorithm task consists of two main steps:
1) Branch metrics and class metrics computation.
2) Comparison of the extension sequence with all the candidate sequences.

The first step is equivalent to the starting step of the Viterbi algorithm. The second step, which is the most computationally intensive, needs a number of comparisons that depend implicitly on the SNR through (8). Thus, the number of comparisons made by the algorithm is an rv. This aspect makes the proposed algorithm functionally similar to the class of sequential algorithms. The complexity of the second step of the algorithm has been measured through the average number of comparisons required for a given SNR. This number can be estimated through simulation, up to a SNR corresponding to error probabilities of the order of \( 10^{-5} \). To go beyond this limit, in the next section we derive an analytical approximation of this parameter.

The complexity of the Viterbi algorithm is independent of the SNR and will be measured in terms of the numbers of comparisons between path metrics, which is equal to \( N_e(|F_1| - 1) \). This value will be reported in the comparison curves presented in Section VII. As to the Fano sequential algorithm [3], we have already pointed out that its complexity is highly dependent on the SNR, so that we have evaluated it as follows. Let \( N_F \) and \( N_B \) be the average number of forward and backward steps performed by the Fano algorithm. Then the algorithm complexity has been estimated by simulation as
\[
C_F = (N_F - 1)|F_1| + N_B.
\]
A. Analytical Results

The average number of comparison $\bar{N}$ can be computed considering separately the terms $\bar{N}_i$ relative to each of the tables $\mathcal{M}_i$ stored

$$\bar{N} = \sum_{i=1}^{p} P_i \bar{N}_i.$$  

In this expression, $P_i$ is the probability of accessing the table $\mathcal{M}_i$, event which happens when the parameters $\Delta_1$ or $\Delta_2$, defined in (6) and (9) for the thresholds $T_1$ and $T_2$, is greater than zero. This probability can be approximated, assuming that the ML sequence coincides with the transmitted one, by taking the union bound over all the distances between the signals belonging to $F$, and the identity signal zero

$$P_i \approx \sum_{j=1}^{|F_i|} \frac{1}{2} \text{erfc} \left( \frac{d_j}{\sqrt{8\sigma}} \right).$$

In order to compute the terms $\bar{N}_i$, we derived the probability density function (pdf) of the thresholds $T_1$ and $T_2$. In the following, we only report the results obtained for the threshold $T_2$. The quantity $\bar{N}_i$ can be computed as

$$\bar{N}_i = \sum_{h=1}^{h_{\text{max}}} \int_{0}^{\infty} N_i^{(h)}(y) f_T^{(h)}(y) \, dy$$

where $f_T^{(h)}(y)$ is the pdf of the threshold $T_2$ (a function of $h$),
conditioned on the event $\Delta_2 > 0$, and $N^h(y)$ is the number of error sequences belonging to the table $\mathcal{M}_i$ with length $h$ satisfying

$$D^2 \leq y.$$ 

For $L \times$MPSK group codes the threshold $T_2^2$ depends on the scalar product $A' \cdot R'$ and on the parameter $\Delta_2$. Since the sequence $A'$ is the ML sequence, the scalar product can be lower bounded by

$$A' \cdot R' \geq T' \cdot R'$$

where $T'$ is the transmitted sequence.

Owing to the spherical symmetry of the multivariate Gaussian distribution, the noise vector can be expressed in terms of an orthonormal basis having one axis parallel to $T'$ and the remaining $2hL - 1$ axes orthogonal to $T'$. Define with $n||$ and $n_\perp$ the rv's representing the noise components parallel and orthogonal to $T'$. Then

$$T' \cdot R' = hL + n|| \sqrt{hL}$$

$$|R'|^2 = \left(\sqrt{hL} + n||\right)^2 + |n_\perp|^2.$$ 

With these assumptions, the square of the threshold $T_2^2$ can be upper bounded by

$$T_2^2 \leq 4 \left[ hL + \frac{(hL + n|| \sqrt{hL})(hL + n|| \sqrt{hL} - \Delta_2)}{(\sqrt{hL} + n||)^2 + |n_\perp|^2} \right]. \quad (10)$$

We approximate the pdf of $T_2^2$ with the pdf of the right hand side (RHS) of (10). To compute it, we first condition it with respect to the rv's $n||$ and $\Delta_2$. Performing the rv transformation in the RHS of (10), we obtain the pdf of $T_2^2$ as a function of the pdf of $|n_\perp|^2$

$$f_{T_2^2}(y|n||, \Delta_2) \approx \frac{4(hL + n|| \sqrt{hL})(hL + n|| \sqrt{hL} - \Delta_2)}{(4hL - y)^2} \left| f_{|n_\perp|^2}(x_0) \right|$$

where $f_{|n_\perp|^2}(\cdot)$ is the pdf of the rv $|n_\perp|^2$, which, owing to the meaning of $|n_\perp|^2$ is x-square with $2hL - 1$ degree of freedom and

$$x_0 = \frac{4(hL + n|| \sqrt{hL})(hL + n|| \sqrt{hL} - \Delta_2)}{4hL - y} - (\sqrt{hL} + n||)^2. \quad (11)$$

Then, we numerically average over the conditioning rv's $n||$ and $\Delta_2$

$$f_{T_2^2}(y) = \int_{-\infty}^{\infty} f_{T_2^2}(y|n||, \Delta_2) f_{n||}(x) f_{\Delta_2}(y) \, dx \, dy.$$
where \( f_{\eta_1}(x) \) is a zero-mean Gaussian pdf with variance depending on the SNR, and \( f_{\Delta_2}(y) \) is the pdf of the rv \( \Delta_2 m(\Delta_2) \), because of the hypothesis we made that \( \Delta_2 \) be greater than zero. This density function is easily derived from a transformation operated on the Gaussian rv \( \Delta_2 \).

To validate the approximations made in the analytical evaluation of the statistics of the threshold, we compare the analytical curves with those obtained through simulation. This has been done for a TCM code with 32 states using a \( 2 \times 4 \) PSK constellation [4], and the results are shown in Fig. 6. They fully confirm the validity of the analytical approximations, and allow a complexity comparison with the Viterbi algorithm also for values of SNR's not achievable through simulations.

VI. ANALYTICAL BOUNDS TO THE ERROR PERFORMANCE

The proposed algorithm incurs decoding errors because of two different phenomena, one related to the code

Fig. 8. Simulation results for the \( 2 \times 4 \) PSK codes with \( d_{\text{max}} = 12, 14, 16, 18, \) and 20.
characteristics and the other dominated by the parameter $d_{\text{max}}$.

The first situation arises when the ML sequence is different from the transmitted one, so that the ML decoding is wrong. The second one happens when the algorithm fails to track the ML sequence.

As a consequence, we can write the error event probability as the sum of two terms

$$P_e(e) = P_s(e) + P_f(e)$$  \hspace{2cm} (12)

reflecting the two different causes of error, that can be computed separately.

For both terms, we can obtain an upper bound based on the union bound computed on a suitable set of sequences. In fact, it is well known that for the first term $P_s(e)$ (the standard error event probability for ML decoding), the sequences appearing in the union bound are all the sequences that diverge at some time from the identity sequence 0 and remerge again into it after a certain number of steps

$$P_s(e) = \sum_{d_j = d_{\text{max}}}^{\infty} N(d_j) \text{erfc} \left( \frac{d_j}{\sqrt{8\sigma}} \right).$$  \hspace{2cm} (13)

On the other hand, as all sequences with distance smaller than or equal to $d_{\text{max}}$ from the transmitted one are considered in each decoding step, the probability of loosing the ML sequence equals the probability that the ML sequence be any sequence with distance greater than $d_{\text{max}}$

$$P_f(e) = \sum_{d_j = d_{\text{max}} + \epsilon}^{\infty} N(d_j) \text{erfc} \left( \frac{d_j}{\sqrt{8\sigma}} \right)$$  \hspace{2cm} (14)

where the term $\epsilon$ simply means that the sum starts from the distance immediately greater than $d_{\text{max}}$. This sum is extended over all sequences that diverge from the identity sequence, irrespective of the fact that they remerge or not into it. This makes the multiplicities $N(d_j)$ in (14) greater than those in (13).

As the asymptotical behavior of (12) is dominated by the least among the first terms in the summations (13) and (14), it becomes evident that choosing $d_{\text{max}}$ greater than or equal to $d_{\text{free}}$ guarantees the asymptotical optimality of the algorithm.

However, due to the high multiplicities of (14), at medium and low SNR's a nonnegligible loss can occur depending on the choice of $d_{\text{max}}$.

In Fig. 7, we plot the two contributions to the error probability, together with their sum in (12). For comparison, also the results of the simulation are reported, for the same code considered in Fig. 6.

It can be noticed that at low SNR's the bounds diverge from the simulation results owing to the high multiplicities that appears in (14). This is a standard behavior for these bounds, whose applicability complements that of simulation, for medium-high SNR's.

Fig. 9. The computational complexity of the proposed algorithm as a function of the memory $\nu$ of the code.

VII. PERFORMANCE OF THE ALGORITHM AND COMPARISON WITH THE VITERBI ALGORITHM

The performance of the proposed algorithm has been compared to that of the Viterbi algorithm in terms of error event probability and complexity in extensive simulation runs.

We have used as test codes the best rate $\frac{1}{2} 2 \times 4$ PSK geometrically uniform codes reported in [5] with 16, 32, 64, and 128 states. Their description is reported in Table I, where the notations are in agreement with those of [5].

The results are reported in Fig. 8, containing the curves of error event probabilities and, on the same plot, the average number of comparisons between sequences and the complexity of the Viterbi algorithm.

A first comment concerns the fact that the complexity of the algorithm is almost independent of the choice of the code, and, instead, closely related to the choice of the parameter $d_{\text{max}}$. This is better seen in Fig. 9, where we plot the average number of comparisons as a function of the constraint length of the code.

Moreover, we can observe that, for all codes, the curve giving the average number of comparisons for $d_{\text{max}} = d_{\text{free}}$ becomes smaller than the complexity of the Viterbi algorithm for low values of the SNR, with a slight penalty in error rate.

VIII. A FURTHER SIMPLIFICATION OF THE ALGORITHM

As we observed the behavior of the algorithm during extensive simulation runs, an anomaly became apparent concerning the thresholds used in the algorithm. In fact, the threshold (8) increases with $h$, and, consequently, the number of sequences examined by the algorithm increases enormously, especially for low SNR's.

To reduce the computational complexity with a negligible loss in performance we have divided the threshold (8) by a
Fig. 10. Error event probability (bit error probability in the case of Fano algorithm) and average number of comparisons for the 32 states code and $d_{\text{max}} = 20$. The curves refer to the proposed algorithm (diamonds), to its improved version, described in Section VIII (white circles), to the Viterbi (continuous line) and Fano (black circles) algorithms.

Factor $K \geq 1$ dependent on $h$ and on the variance of the noise through

$$K = 1 + G_1 h \sigma^2.$$  

Here, the constant $G_1$ is optimized for each code and the variance $\sigma^2$ is estimated from the received data. In Fig. 10 the results obtained using the reduction factor $K$ for the 32 states code are drawn, together with those pertaining to the threshold (8). The error probabilities and complexity of both Viterbi and sequential Fano algorithms are also shown. The gain in complexity with respect to the Viterbi algorithm is quite sensible, ranging from one to almost two orders of magnitude. With respect to the Fano algorithm, the new algorithm behaves better for low values of the SNR's, while losing something for high SNR's. The structure of the new algorithm, however, is more uniform and regular than classical sequential algorithms, lending itself to an easier very large scale integration (VLSI) implementation, which is presently under way for a linear convolutional code with very large constraint length.

IX. CONCLUSIONS

We have proposed a new decoding algorithm applicable to geometrically uniform trellis codes, which exploits their group properties in order to reduce the computational complexity. The complexity of the algorithm is almost independent of the number of states of the TCM scheme, and results from a trade-off with the performance. In essence, the algorithm keeps track of only one path in the trellis, and this makes it similar, in some instances, to sequential algorithms.

The complexity and performance have been compared to the Viterbi algorithm, showing that considerable gains can be obtained, especially using a modified version of the original algorithm.

As a particular case, the algorithm can be applied to the ML decoding of binary convolutional codes. Its applicability and performance for large constraint length convolutional codes is presently being investigated, having in mind a VLSI implementation.

REFERENCES


Sergio Benedetto (SM’90) received the “Laurea in Ingegneria Elettronica” (summa cum laude) from Politecnico di Torino, Italy, in 1969.

From 1970–1979, he was with the Istituto di Elettronica e Telecomunicazioni, first as a Research Engineer, then as an Associate Professor. In 1981, he was made a Professor in Radio Communications at the Università di Bari. In 1981, he was back to Politecnico di Torino as a Professor in Data Transmission Theory at the Electronics Department. He spent nine months in years 1980–1981 at the System Science Department of University of California, Los Angeles, as a Visiting Professor. Active in the field of digital transmission systems since 1970, his current interests are in the field of performance evaluation and simulation of digital communication systems, trellis coding and VLSI algorithms for telecommunications, and optical communications systems. He has coauthored two books in signal theory and probability and rv's (in Italian), and the book Digital Transmission Theory. Englewood Cliffs, NJ: Prentice-Hall, 1987, as well as a hundred papers in leading engineering conferences and journals.

Mr. Benedetto is Area Editor for signal design, modulation and detection for the IEEE TRANSACTIONS ON COMMUNICATIONS.

Guido Montorsi (M’95) was born in Turin, Italy, on January 1, 1965. He received the Laurea in Ingegneria Elettronica in 1990 from Politecnico di Torino, Turin, Italy, with a master’s thesis concerning the study and design of coding schemes for HDTV, developed at the RAI Research Center, Turin. He received the Ph.D. degree in telecommunications from the Dipartimento di Elettronica of Politecnico di Torino, in 1994.

In 1992, he spent the year as visiting scholar in the Department of Electrical Engineering at the Rensselaer Polytechnic Institute, Troy, NY. His current interests are in the area of channel coding, particularly analysis and design of concatenated coding schemes and study of iterative decoding strategies.