Breadth-First Trellis Decoding with Adaptive Effort

STANLEY J. SIMMONS

Abstract—**A new breadth-first trellis decoding algorithm is introduced for application to sequence estimation in digital data transmission. The high degree of inherent parallelism makes a parallel-processing (possibly VLSI) implementation attractive. The proposed algorithm is shown to exhibit an error-rate versus average-computational-complexity behavior that is much superior to the well-known Viterbi algorithm, and which also improves on the popular *M*-algorithm. The decoding algorithm maintains a variable number of paths as its computation adapts to the channel noise actually encountered. Buffering of received samples is required to support this.**

Bounds which are evaluated by trellis search are produced for the error event rate and average number of survivors. Performance is evaluated with conventional binary convolutional codes over both BSC and AWGN channels. Performance is also found for multilevel AM and PSK codes and simple intersymbol interference responses over an AWGN channel. At lower SNR, Monte-Carlo simulations are employed to improve on the bounds and to investigate decoder dynamics.

I. Introduction

TRELLIS structure pervades digital communication signalling. Two of the most common sources of this structure [1] are the application of trellis codes for forward error correction, and the presence of intersymbol interference (ISI) due to nonideal channel responses. So called "coded modulations" [15] also impress a trellis structure on the transmitted signal. The Viterbi algorithm (*V*-alg) [2] is well-known as the optimal (sequence) estimator for trellis-structured signals. Its computational complexity, however, is proportional to the width of the trellis, which is in turn exponential in the memory length of the coder, modulator, or channel response. Viterbi decoding can therefore become impractical with more powerful codes and modulations, or more dispersive channels.

Because of this computational problem with the Viterbi algorithm, alternative reduced-computation algorithms (that are necessarily suboptimal) have been proposed in the literature. These algorithms fall generally into three classes as defined in [3]: depth-first, metric-first, and breadth-first trellis search. Depth-first algorithms search along a single promising sequence, backtracking to explore alternate paths when it seems likely that a wrong turn has been made at some earlier point. Metric-first algorithms rank sequences in contention according to the goodness of fit to the observed (received) signal, and extend and explore the sequence that has the current best metric. Breadth-first algorithms, however, extend many possible sequences during a processing interval and prune contenders according to a discard criterion based on metrics. All contending sequences are of the same length and there is never any backtracking. The Viterbi algorithm belongs to this class.

Paper approved by the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received August 18, 1987; revised December 20, 1988. This work was supported in part by a Natural Sciences and Engineering Research Council of Canada (NSERC) University research Fellowship. This paper was presented in part at the 14th Biennial Symposium on Communications, Queen's University, Kingston, Ont., Canada, May 29–June 1, 1988.

The author is with the Department of Electrical Engineering, Queen's University, Kingston, Ont. K7L 3N6, Canada.

IEEE Log Number 8932066.

The motivation for this work is ultimately to allow high data rates over existing noisy and/or dispersive channels. Since higher data rates imply the necessity for complex codes and/or the presence of increased dispersion, computationally fast decoders are required. To this end, desirable attributes for a decoding algorithm are both reduced computation, and the ability of an implementation to exploit parallelism with suitable (possibly VLSI) structures. Decoding algorithms which are purely breadth-first are inherently parallel, and are therefore of greatest interest here. Examples of VLSI structures that exploit the parallelism of breadth-first decoding are given in [16]–[18].

The well-known *M*-algorithm [3] belongs to the breadth-first decoding class. Like the Viterbi algorithm, however, it performs a fixed number of computations per decoded symbol. This fixed computational load must be tailored for worst case channel conditions; the algorithm cannot capitalize on periods where decoding decisions may be much easier and require less computation. The algorithm proposed in this paper exhibits an inherent adaptation of decoding effort which lowers the average computational complexity at the expense of requiring a buffer and introducing modest decoding delays.

Other proposals for suboptimal breadth-first decoding are evident in the literature, but they tend to be restricted in application to the specific trellises for which they were designed; see, for example, references [4]–[7]. The new algorithm described in this paper is applicable to any trellis-structured signal.

II. Discrete-Time Model

The very general discrete-time communications model of Fig. 1 is assumed. This model may be derived from the standard continuous-time communications model after appropriate matched-filtering and symbol-rate sampling of the received signal. Information digits u_k, chosen from an alphabet of size b, are produced every Υ seconds (for a symbol rate of $1/\Upsilon$) and are assumed independent and equiprobable. Subscript k denotes the time index or symbol interval. The channel symbols x_k are a function of the current input u_k and ν past inputs, $x_k = f(u_k, s_{k-1})$ where s_k is a state uniquely determined by the ν inputs immediately prior to time $k\Upsilon$. The one-to-one mapping between input sequences $\boldsymbol{u}$ and channel symbol sequences $\boldsymbol{x}$ is described by a trellis of width $S = b^{\nu}$ states. Fig. 2 is an example of a simple eight-state trellis. There are b branches out of each state, one per possible input symbol, and each branch has a corresponding channel symbol x_k. Trellis paths are a concatenation of branches. The observation y_k is the sum of the transmitted x_k and noise term n_k where the n_k are assumed to be independent random variables (and independent of the x_k).

Associated with each possible sequence $\boldsymbol{x}$ is a metric; here a maximum likelihood metric is assumed,

$$\Gamma(\boldsymbol{x}, \boldsymbol{y}) = -\log_e p(\boldsymbol{y}|\boldsymbol{x}) = -\log_e \prod_k p(y_k|x_k)$$

$$= -\sum_k \log_e p(y_k|x_k) = \sum_k \gamma(x_k, y_k) \qquad (1)$$

where the probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ breaks into a product of terms due to the assumed independence of the additive n_k. The path

0090-6778/90/0100-0003$01.00

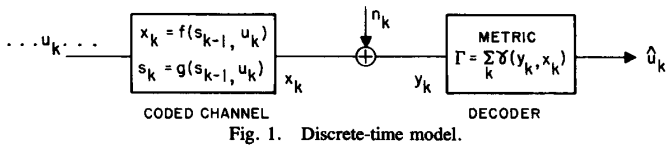


Fig. 1. Discrete-time model.

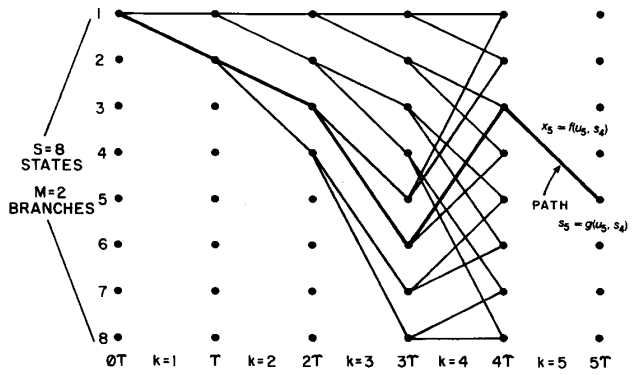


Fig. 2. Simple eight-state trellis.

corresponding to sequence x then has a cumulative metric Γ that is the sum of the metrics of its branches.

III. Proposed Decoding Algorithm

An optimal ML decoder chooses the path through the trellis from initial state s_0 having the best (minimum) metric as defined above. The Viterbi algorithm recursively implements this exhaustive search by maintaining one survivor path to each state in the trellis at each time $k\Upsilon$. That survivor is the path with minimum metric over all paths that can reach the state. We can avoid this exhaustive approach by discarding paths at any depth k in the trellis that have a "high" metric and are therefore unlikely to be the transmitted path. Since path metrics will accumulate with time, we cannot simply compare path metrics to a fixed threshold. Introducing a threshold with a bias that increases with depth in the trellis is one possibility. Such a bias is incorporated into path metrics [8] for depth-first and metric-first decoding [3] to allow comparisons between paths at different depths. Such an approach does, however, require knowledge or estimates of the channel signal-to-noise ratio. Also, the bias is selected for average channel noise; when noise is high, all paths will tend to increase their metrics and look poor. A better approach may be to sample the prevailing noise conditions by using the metric of the best path in contention. The resulting simple discard rule is

- discard path i at time $k\Upsilon$ if $\Gamma_i - \Gamma_B > T$

where Γ_B is the metric of the best path at depth k, and T is a fixed threshold. Note that the choice of the lowest metric for subtraction maximizes the metric differences and will therefore minimize the number of paths which survive the rejection tests.

To complete the decoding algorithm, it is necessary to specify how a decoded output is obtained from the list of survivor paths. Where the paths all merge to a common history (see Fig. 3) the decoder output path is simply the common section. More precisely, the common branch symbol for time $(k-L)\Upsilon$ is released where L is the decoding depth. It is possible that the paths will not have merged by this depth, and this may be quite common when higher rejection thresholds are used. Clearly, the best strategy in this situation is (as in a Viterbi algorithm implementation) to take the decoder decision path branch to be that belonging to the path in storage having the lowest metric, that is, the path of highest likelihood. All unmerged paths are then rejected.

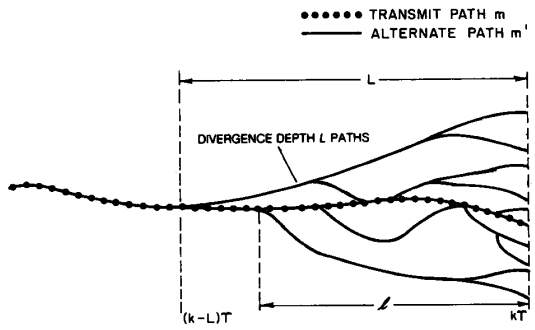


Fig. 3. Typical contender paths.

A formal statement of the recursive algorithm for time $k\Upsilon$ is the following.

1) Extend each survivor from $(k-1)\Upsilon$ into b contenders, concatenating the new symbol to the path history and updating the accumulated path metric of each contender.
2) Find and mark the contender having best metric Γ_B.
3) Subtract Γ_B from the metric of each contender and compare to threshold T, marking those that fail the test for rejection.
4) Mark for rejection all contenders whose path history symbols at $(k-L)\Upsilon$ do not agree with that of the best path.
5) Purge all contenders marked for rejection to form the new survivor list at time $k\Upsilon$.

Note that the subtraction of Γ_B normalizes all path metrics so that metric overflow is really not a problem. Note also that since path states do not enter into the path-rejection processing, it is possible that distinct survivors may share a common state at time $k\Upsilon$. In this mode of operation, the decoder views the trellis as a tree, and is unaware of the concept of path states. This mode of operation has been deliberately chosen. It would have been possible to implement an additional rejection step to purge all but the path of lowest metric in any group of paths that share a common state, as does the Viterbi algorithm. For example, to discover which contender paths have common states, an efficient approach would be to sort all the paths by state. Such sorting, however, represents extra work, and

requires extra hardware in order to be accomplished with any sort of parallelism. In addition, simulations have shown that only marginal reductions in the average number of survivors can be gained by performing the extra state-duplication purging.

Note that step 4) above is eliminated from some reduced-computation algorithms, the justification being that its inclusion gives only negligible performance improvement. Step 4) cannot be eliminated here however, as it compensates for not checking for state duplications in the survivor list. If such paths are not eventually eliminated, they can cause the size of the survivor list to grow continually, with no corresponding increases in algorithm error-rate performance. Since any paths that rejoin to a common state must have diverged at some earlier time, step 4) will discover this divergence and discard the poorer of such paths. State duplicating paths are therefore still eliminated, but only after some delay needed for their initial divergence to appear in the path histories at depth L. It should be emphasized that all bounding and simulation results shown later include this effect, as it is an inherent part of the algorithm defined above.

There is another important consideration. Because the survivor number S_T is variable, it may at times tend to grow quite large. There must be some limit enforced on S_T in a practical implementation. A "soft" limit is proposed where iterative reprocessing with tightened thresholds is performed until the tentative number of survivors is equal to the limit S_L. The most efficient exact implementation of this soft-limiting is a binary search for the threshold that gives exactly S_L survivors. An inexact but simpler implementation, and the method adopted for the results shown later, is to reduce the threshold by 10 percent at each reprocessing step until S_L or fewer paths remain. No matter what method is used, soft-limit reprocessing will increase the average computational load. As long as S_L is sufficiently high, however, this limiting operation will occur only infrequently, and can contribute negligibly to the average computation. This is evident in the later discussion of the results.

IV. Performance Analysis

The error-rate measure of interest is defined to be the time-average per-symbol probability of an error event starting. The average number of survivors is taken as a measure of computational complexity. Upper bounds on these quantities are produced. The maximum number of survivors is not a relevant criterion where survivor number limiting will ultimately be applied. In addition, for some signal structures the upper bound on this maximum is b^L, which will almost always be too large to be useful.

An error event will be defined by the divergence of the decoder decision path from the transmitted path m (m stands for message). The probability of this occurring is of interest here; the number of symbol errors and the length of the error event are not of concern. It is assumed that the decoder knows the initial state of the coder or channel.

Rejection of the transmitted path is a necessary and sufficient condition for an error event to begin. Rejection of the transmitted path can occur only due to a failed threshold test or a forced decision at decoding depth L. Typical paths in contention are shown in Fig. 3. An invocation of stationarity and ergodicity coupled with a standard union bound argument (see Appendix A) allows the following ensemble-average upper bound on rejection probability P_R:

$$P_R \le \sum_m P(m) \left[\sum_{\{m'_l, l=1,L\}} P_t(m'|m) + \sum_{m'_L} P_f(m'|m) \right] \quad (2)$$

where all transmitted paths m, and alternate (contending) paths m', are confined to the interval $[0, L\Upsilon]$; l represents divergence depth; $P(m)$ is the probability of message m; and P_t and P_f are used to denote threshold-test and forced-decision probabilities for a single alternate m'. This is simply a summation over all possible alternate paths that could become the best path and lead to rejection of the transmitted path. The probability of error in a block of N symbols would then be given by NP_R. The defining probabilities for P_t and P_f are

$$P_t(m'|m) = P[\Gamma_m - \Gamma_{m'} > T] \text{ and } P_f(m'|m) = P[\Gamma_m - \Gamma_{m'} \ge 0]. \quad (3)$$

Explicit expressions for P_t and P_f for the BSC and AWGN channels can be found in Appendix B. A very similar form can be constructed to bound the Viterbi algorithm error-rate. For the Viterbi algorithm bound, only those paths m' that rejoin the state of the transmitted path at time $L\Upsilon$ are counted, and these paths are considered to be involved in a metric-difference test with the transmitted path with an effective threshold T of zero.

The above bound result is based on a long-time average. For the proposed algorithm, there will be a transient startup period for $k < L$ in any real transmissions where the probability of an event starting is somewhat lower than the long-time average. This is because of the information provided by the known start state; fewer alternate paths are initially in contention. Similarly, if known information symbols u_k are inserted into the data stream (a useful strategy described later), the decoder using this information lowers P_R for some intervals. Ignoring these two effects clearly produces an overbound on the average error event rate P_E.

The following bound on average number of survivors, *conditioned on the transmitted path m being on the contender list*, can also be produced (see Appendix A):

$$\bar{S}_T \le \sum_m P(m) \sum_{m'} P_s(m'|m) + 1 \quad (4)$$

where $P_s(m'|m)$ denotes the conditional survival probability for m' and is given by

$$P_s(m'|m) = P[\Gamma_{m'} - \Gamma_m \le T]. \quad (5)$$

The bound's first term is simply a summation over all alternate paths with weighting by the probability that the path would survive a threshold test in which Γ_m replaces Γ_B. Explicit expressions for P_s for the BSC and the AWGN channel can be found in Appendix B.

Conditioning the bound on the transmitted path being on the contender list is not a drawback if the primary concern is the probability of an error event *starting*, as is assumed. Naturally, the speed of the decoder implementation will be tailored for the average survivor number occurring in the absence of errors, that is, with the transmitted path present. If the transmitted path has already been rejected, the bound may then not hold, but the behavior of the survivor number in this case is unimportant. Even if S_T should grow uncontrollably and quickly reach an enforced limit S_L, the decoder action can only result in a longer event, not a new event. It should also be mentioned here that the above bound has been developed assuming the decoder does not enforce a limit on the number of survivors. In the presence of limiting, the average number is actually found to decrease, as might be expected, at the expense of incurring extra computations needed to perform the limiting.

V. Bound Evaluations and Simulations

The bound expressions above require for a given transmitted path m that all alternate paths m' of length L symbols be explored. Since the constituent probabilities depend only on distance $d(m, m')$ [Euclidian or Hamming] for the AWGN or BSC channels, this evaluation may be done by counting paths at each discrete value of distance $d(m, m')$, and summing the contributions over all these distances. This path distance enumeration can be done without redundant path tracing by combining path counts where paths enter a common trellis state, and share a common distance $d(m, m')$. A form of trellis "search" to depth L accomplishes this.

One immediate problem is that the number of discrete values of $d(m, m')$ may be very large. To deal with this problem, a distance boundary d_B was defined that separated paths into "low" and "high"

distance categories. Once paths accumulate distance in excess of d_B, they are graduated into the high distance group. For each of the low distances, values for P_t, P_f, and P_s are calculated and applied to the path counts, but the high-distance path contributions are combined into a single Chernoff bound with bound parameter $\lambda = 1/2$. The Chernoff bounds for these probabilities are given in Appendix B. If the $\lambda = 1/2$ bounds had been applied uniformly to all paths m' regardless of distance $d(m, m')$, the trellis search procedure would have been greatly simplified. In fact, it would have effectively become an efficient method for evaluating a transfer function bound [1] of the same form as those regularly employed in the literature for the Viterbi algorithm. The reason that this was not done is that the $\lambda = 1/2$ bound is too weak for the lower distance paths that take part in the proposed algorithm. For the higher path distances however, the $\lambda = 1/2$ bound is tight, and may be employed. In practice, the boundary distance d_B was chosen sufficiently high that the contribution of the high-distance paths was much smaller than that of the lower distance paths, yielding as tight a bound as possible.

For trellises whose distance profile is independent of the transmitted path, a single search with the all-zeros as transmitted path m produces the final bound. For trellises where the result depends on the particular transmitted path, an average over all such paths is needed. An exact computation can be performed based on a super-state (or "pair-state") [9] description of the trellis involved. The number of super-states is the square of the number of normal states, and computational and storage requirements grow rapidly with channel or code memory. An alternative approach, and the one used here, is to simply form a probabilistic estimate of the bound based on combined time and ensemble averaging. To accomplish this, a random path m is traced through the trellis to depth $L + N$ symbols, and the per-symbol average bounds evaluated over these N symbols. Additional random paths are then traced and evaluated, redoubling the total observed paths at each iteration to $2, 4, 8, 16 \cdots$ until there is less than a 5 percent change in a cumulative bound average. This approach, although not exact, produces bounds in which one can be quite confident while avoiding the exhaustive super-trellis computation. This approach was verified against the exact super-state method for codes whose constraint length was low enough to permit this.[1]

Finally, some of the redundancy in the union bound on P_R could be easily removed during the evaluations. This type of redundancy occurs where paths m' define error regions in the signal space that are completely covered by error regions already counted at a different time $k\Upsilon$. This includes paths whose last symbol x'_k is the same as symbol x_k of the transmitted path, as well as paths that have rejoined the state sequence of the transmitted path for any duration. The redundancy elimination is accomplished by not adding in P_t contributions from these paths.

Monte-Carlo simulations of the Viterbi, M-algorithm and the proposed T-algorithm were carried out. These simulations provided tighter results at the lower SNR's where the bounds are weaker, and allowed investigation of decoding dynamics. For the M-algorithm simulation, no purging of state-duplicating paths was implemented. This creates more of a common ground for comparison, and only marginally increases the error rate compared to an algorithm in which purging is used.

The simulations were carried out using vector space representations. For example, with 4-level amplitude modulated codes, the sequence of transmitted levels can be simply interpreted as a vector with one dimension per baud interval (for PSK modulation there are two dimensions per baud). Any other contender sequence can be similarly represented. To simulate white Gaussian channel noise, an independent noise sample is added to each coordinate of the transmitted vector, where the variance of this zero-mean noise sample is equal to $N_o/2$, the height of the two-sided noise power spectral density. The noise samples are generated using a proven random-number generator. This simulation method implies that we are assuming an ideal receiver that uses matched filtering. Squared Euclidian distance is calculated as the sum of the squares of the differences between the coordinates of the received vector and the signal vector of interest. For the BSC simulations with rate 1/2 codes, there are two new dimensions per baud, with each coordinate of the received vector constrained to the set $(+1, -1)$. Hamming distances are used in this case.

The signal-to-noise ratio for the AWGN channel codes is defined by $\text{SNR} = 10\log_{10}(d^2_{\min}/2N_o)$ where $d_{\min}$ is the minimum distance between symbols in an uncoded system having equal average received power. This also means that the coded and uncoded systems transmit the same average energy per source bit, since their bit rates are the same; the coded system bandwidth is only greater than the uncoded system in the case of the binary convolutional codes. Note that the above expression for SNR also applies to the intersymbol interference cases, except that $d_{\min}$ in these cases is the minimum distance between distinct paths in the ISI trellis. This definition of SNR allows the error-rate for different codes/ISI to be shown on the same scale for a given SNR; conversion to an E_B/N_o SNR measure is trivial.

It should be mentioned that for all results, decoding depth L was selected large enough that there was a negligible contribution from forced decisions involving paths m' that are completely unmerged with transmitted path m for the full L symbols. That is, virtually no improvement in performance can be found by increasing L further (the required value of L was found by repeated trials). This applies both to the bound results and the simulation results.

For the purposes of simulation, the transmitted symbols were grouped into blocks of 200 symbols, with the last few symbols in each block chosen to drive the state of the transmitted path back to the "all-zeros" state. This periodic state-forcing is exploited by the decoder to limit the length of error events, as is discussed later. Finally, for all simulation results shown in the next section, the soft limit S_L was set high enough that limiting did not occur in any block that was free of errors. The effect of a lower soft limit is discussed at the end of the next section.

VI. Error-rate Versus Complexity: Results and Discussion

To begin, we will consider a well-known trellis structure, that is, the one formed by standard rate-1/2 binary convolutional codes. For this purpose, the ODP codes of [11, Table 12.1] were used. Our definition of constraint length ν is consistent with the definition in [11], namely, that the number of states is given by 2^ν. Algorithm performance has been evaluated with these codes assuming two different channels: a binary symmetric channel (BSC), and an additive white Gaussian noise channel.

It may help to restate the threshold rule in the context of the BSC. The algorithm actually uses Hamming distances for working metrics and thresholds in this case. Proceeding breadth-first, we evaluate the Hamming distances of the contender paths, and find the path with the lowest Hamming distance Γ_B. Now any paths whose Hamming distances exceed Γ_B by more than T_H are rejected, where Hamming-distance threshold T_H takes on integer values.

The top curve of Fig. 4(a) shows the behavior of the proposed threshold algorithm with a constraint length $\nu = 7$ code assuming a binary symmetric channel having channel crossover probability $p = 0.01$. The results on this figure are all from upper bound calculations. Note that as the threshold is increased, the error event rate decreases due to a lower probability of rejecting the transmitted path, but this error-rate improvement comes at the expense of a higher average number of survivors. At high thresholds, however, the error-event rate saturates (as it must) at the error rate that would be realized with optimal Viterbi decoding. The Viterbi-algorithm occupies a single point on this figure where its average number of survivors is simply the number of states, here given by $2^7 = 128$.

Once we get close to saturation, there are rapidly diminishing returns in increasing the threshold further. At these high thresholds, the probability of discarding the transmitted path due to its failing a threshold test is comparatively negligible. Virtually all of the errors

[1] It appears that the new results of [10] could have been useful here with the code trellises (but not the ISI trellises).

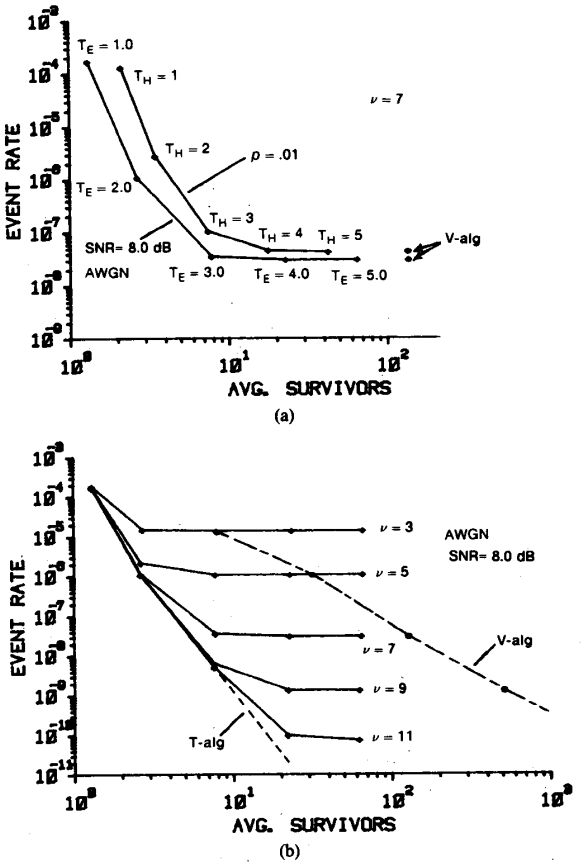


Fig. 4. Error-rate versus complexity bounds with $R = 1/2$ binary convolutional codes: $\nu = 7$ code on BSC and AWGN channel (a), and code family with AWGN at 8.0 dB (b).

are instead being caused by the forced decisions that must be taken when paths are not merged at depth L. In fact, as $T \to \infty$, the decoder decision paths will become identical for the Viterbi algorithm and the T-algorithm. In that limiting situation, all paths would survive the threshold tests, and the released decoder symbol would be determined solely by the criterion of using the depth-L symbol of the overall best path. Since, by definition, the Viterbi algorithm always releases the depth-L symbol of the overall best path, the decision paths for the two algorithms would necessarily become identical.

The lower curve of Fig. 4(a) is for the same code but now for the AWGN channel, and at an SNR that is about 2.5 dB lower, illustrating the gain realized by the use of soft decoding metrics. In this case, Euclidian squared-distance metrics and thresholds T_E are employed. Fig. 4(b) now shows the algorithm's behavior for the same five threshold values applied now to a set of codes of increasing constraint length that includes the $\nu = 7$ code of Fig. 4(a). The lower curve of Fig. 4(a) can be identified as the middle curve of Fig. 4(b). The threshold values have been left off the figure for clarity, but are readily identified by comparison to this $\nu = 7$ curve. The plotted ticks that are virtually in vertical line with the $\nu = 7$ ticks occur at the same threshold value. This shows the first interesting result: for a given threshold value, the average number of survivors is virtually independent of code constraint length.

The second important point is that as long as the curves are not flattening out approaching saturation at the VA limit, the error-rate performance is also virtually independent of the code constraint length. It depends only on the threshold value employed. This can be seen most clearly in the almost perfect overlap of the initial portions of the curves for the different codes. We can therefore identify the curve

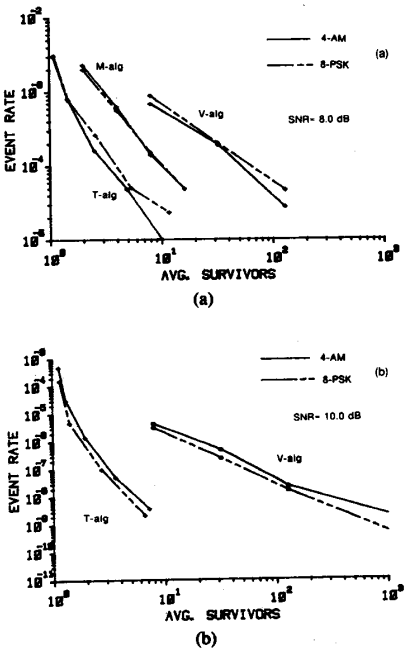


Fig. 5. Error-rate versus complexity for T-alg, M-alg, and V-alg with 4-AM and 8-PSK codes: simulation at 8.0 dB (a), and bounds at 10.0 dB (b).

that follows the lower boundary of this set of curves (suggested by the dotted extension in the figure) as the *error-rate versus complexity characteristic* for the threshold algorithm applied to this family of codes. Also shown is a dashed curve connecting the performance points for Viterbi algorithm decoding over the set of codes. Some liberty has been taken in doing this, since it is not possible to be operating between the discrete Viterbi algorithm points. The dashed curve is simply to show the trend of Viterbi algorithm complexity versus error-rate. It is now evident that the error-rate versus complexity characteristic of the T-algorithm is far superior to that of the Viterbi algorithm. For any target error rate, we can find a threshold for the T-algorithm that will achieve that target error rate but with an average number of survivor paths that is much less than that required by the Viterbi algorithm to achieve the same rate. In Fig. 4, this reduction is in excess of one order of magnitude at an error rate of around 10^{-9}. It should also be stated here that the average number of survivors is a perfectly accurate measure for comparing the computational requirements of the two algorithms. Since both perform identical survivor-path extension processing, and the number of metric comparisons per contender path is the same, the average number of survivor paths is a correct proportional measure.

In Fig. 5, we stick with the concept of an algorithm's error-rate versus complexity characteristic to compare the T-algorithm, the M-algorithm, and the Viterbi algorithm applied with different codes. For this purpose, Ungerboeck 4-AM codes [12, Table I], and Ungerboeck-like 8-PSK codes [13, Table V.3][2] were chosen to demonstrate the algorithm's general applicability. Again, the number of states in the code is given by 2^{ν}.

Fig. 5(a) shows the performance curves at a low SNR, Fig. 5(b) shows them at a higher SNR. Here again, the values of the thresholds used at the plotted points have been left off the figure both for clarity and to make the point that what is important here is an error-rate versus complexity *curve* that is characteristic of the algorithm and the particular family of codes (i.e., 4-AM or 8-PSK) to which it is applied. If a different set of threshold values had been chosen for

[2] These codes were used as they improve on the original Ungerboeck codes at the higher constraint lengths.

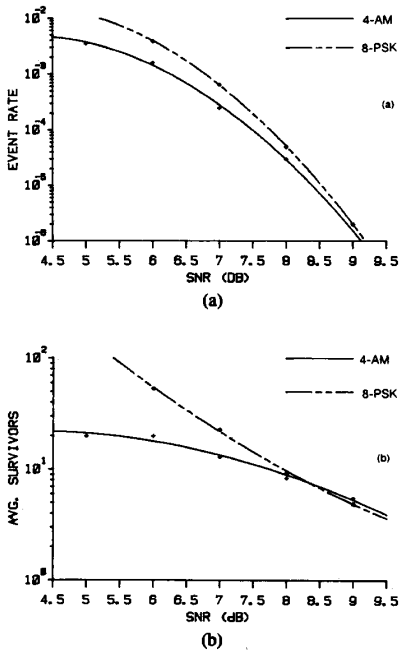


Fig. 6. T-alg behavior (simulation) for changing SNR with $\nu = 10$ 4-AM ($T_E = 16.0$) and 8-PSK ($T_E = 1.6$) codes: error event rate (a), and average number of survivors (b).

the evaluations, the same curve would result, the ticks would just be in different positions on that curve. For information only then, the 4-AM T-alg curve at both SNR's was produced by using the set of thresholds ($T_E = 4.0$, 8.0, 12.0, 16.0, 20.0) with the $\nu = 10$ 4-AM code whose symbols come from the set ($\pm 3, \pm 1$). The 8-PSK T-alg curves were produced using the set of thresholds ($T_E = 0.4$, 0.8, 1.2, 1.6, 2.0) with the $\nu = 10$ 8-PSK code whose symbols are unit-energy phasors spaced in phase by $2\pi/8$.

At the lower SNR, the results are all from Monte-Carlo simulation. Note that the simulations were performed for blocks of data of length 200 symbols (the reason for this is discussed later), and the average number of survivors shown does not include blocks which contain decoding errors. At the higher SNR where simulation is not practical, the upper bound results are used throughout. Note that the M-algorithm could not be included at the higher SNR as there are no upper bounds known for its error-rate performance. That the T-algorithm achieves significant complexity reduction gains over the Viterbi-algorithm is again apparent from Fig. 5. The gains are even more pronounced at the higher SNR (where path discard decisions are easy for a greater portion of the time). Note that the T-algorithm also improves on the M-algorithm. In this case, exact quantitative computational comparisons cannot be made directly since the M-algorithm actually involves extra sorting computations which increase its decoding effort per survivor path. The relative merit of the proposed algorithm is, however, clear.

To this point, the performance of the T-algorithm has not been shown on a familiar error-rate versus SNR plot. This has been done deliberately, as the representation of Figs. 4 and 5 (once accepted) makes the comparisons among the algorithms absolutely clear, and avoids having to compare among multiple figures. In fact, one can find a performance versus SNR curve by interpolating between plots like Fig. 5(a) and (b). However, as the author feels obliged to show at least one such curve, Fig. 6(a) is offered. For this figure, $\nu = 10$ 4-AM and 8-PSK codes have been used, with the threshold T_E selected just below the point at which the error rate would begin to saturate at the V-alg performance limit. This demonstrates that at a fixed threshold choice, the performance degradation with decreasing SNR is graceful, producing a curve whose form is similar to the well-known curve for the Viterbi algorithm.[3] Also shown in Fig. 6(b) is the behavior of the number of survivors as a function of SNR. Quite naturally, the average number of survivors decreases as SNR increases, which is consistent with the algorithm's ability to readily identify and discard incorrect paths when noise is small.

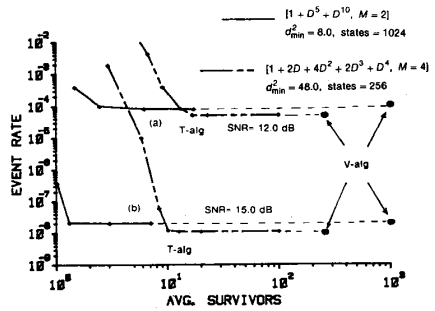


Fig. 7. Error-rate versus complexity for two simple ISI responses at 12.0 dB (a) and 15.0 dB (b).

Finally, Fig. 7 is presented to demonstrate application in the case of a fixed trellis (as opposed to the earlier coded cases where we were free to choose the code constraint length). Results are shown for two representative ISI responses at two SNR's. Here, all results are from upper bound calculations which are very tight. Note that for a given ISI response and SNR, the performance of the Viterbi algorithm is depicted by a single point on the figure, where the average number of survivors is just the number of states. For the T-algorithm, as before, we have a performance curve. This time the curves have been created using the set of thresholds ($T_E = 1.0$, 2.0, 3.0, 4.0) for the $[1 + D^5 + D^{10}, M = 2]$ case, and ($T_E = 4.0$, 8.0, 12.0, 16.0) for the $[1 + 2D + 4D^2 + 2D^3 + D^4, M = 4]$ case. Note that for the former ISI case, the T-algorithm can achieve an error-rate performance virtually identical to that of the Viterbi algorithm with only 1 or 2 survivor paths on average, instead of the 1024 survivors needed by the Viterbi algorithm.[4] The savings for the other ISI case is still almost two orders of magnitude. This shows that it is possible to come very close to optimal (V-alg) performance at a greatly reduced complexity when the signal space is inefficiently packed, as occurs with the two simple intersymbol interference channels tested.

VII. Decoding Dynamics

A variable computational load arises from the variation in the number of survivors kept at each symbol interval; a buffer for the received channel samples is needed to exploit the low average load. An analysis of the survivor-number and buffer-backlog processes seemed very difficult, so simulations were performed at the lower SNR's where the survivor numbers are highest. Computational requirements for each interval are taken to be directly proportional to the actual number of survivors.

Fig. 8(a) illustrates typical behavior for a representative code trellis when decoder capability is set at twice the *average* computational requirement. This behavior is typical over a wide range of codes and illustrates that the low average decoding load can in fact be exploited without excessive buffer backlogs and decoding delays. The relationship between buffer backlogs and survivor dynamics can be seen more clearly in the time scale expansion of Fig. 8(b). When noise is not large, the survivor number hovers close to the average value, and the decoder uses its excess capability to keep the received

[3] Note that algorithm comparisons based on such a representation would be difficult at best. If we are to add a V-alg curve, what constraint length should we use? If we wish to add an M-alg curve, what value of M should be used?

[4] Note that 1024 states applies to a single Viterbi decoder; a reviewer has pointed out that this particular response can be decomposed into 5 interleaved channels with a separate 4-state Viterbi decoder for each. This (unfortunately somewhat degenerate) response was selected simply to show that long memory length responses are easily handled by the T-algorithm.

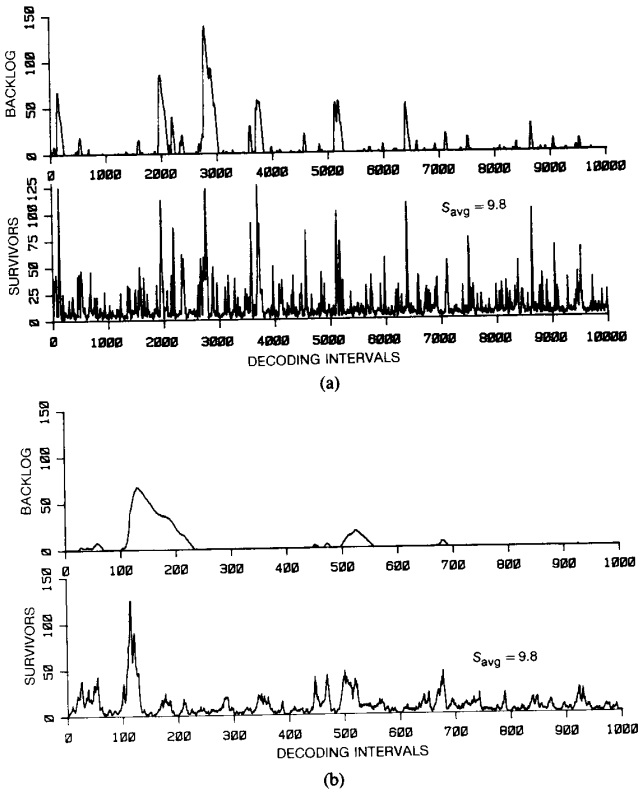


Fig. 8. Typical decoder dynamics (a) and time-scale expansion (b) [actual code used is 4-AM, $\nu = 10$ at SNR=8.0 dB, $T_E = 80$].

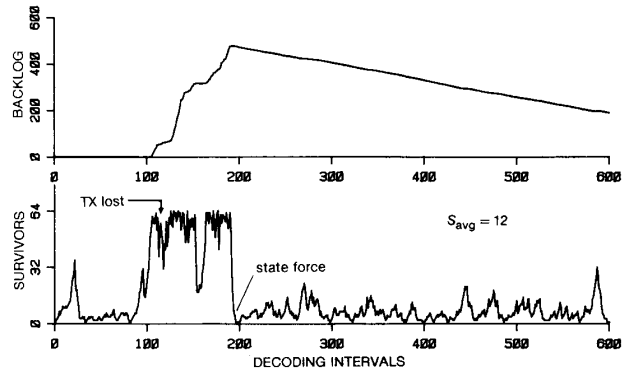


Fig. 9. Decoder dynamics with loss of transmitted path (code of Fig. 8 with SNR = 7.5 dB, $S_L = 64$).

sample buffer cleared. When one or several larger noise samples occur, all path metrics tend to climb and the decoding algorithm responds with an increase in the number of survivor paths, which tends to be sustained over several symbol intervals. The decoding algorithm, in effect, uses this extra observation time to establish which paths may then be safely discarded. Computational demands now outpace decoder capability, and received samples accumulate in the buffer. Once the noise event has passed, it takes some time to clear the computational backlog. Also note that the periodic state-forcing employed in the simulations drives the number of survivors back to just one at the end of every 200-symbol block.

What is being shown here on the horizontal axis of Fig. 8 is the symbol interval that the decoder is processing. These decoding intervals take varying amounts of time to complete, and ther duration only corresponds *on average* to the fixed duration of a transmission baud. The backlog simply shows how many symbols the decoder is behind the transmitter. This explains why the figures do not show the buffer backlogs immediately dropping to zero when state-forcing occurs at the end of each block. The backlog represents unprocessed symbols that we simply cannot discard.

Fig. 9 shows the behavior when the transmitted path is lost and an error event begins. Note that the number of survivors gets very large, and the buffer backlog grows steadily. Soft-limiting to 64 survivors is used to show behaviour under realistic conditions. A practical implementation may require aborting the decoding of such a block and flushing the corresponding portion of the received-sample buffer (the whole block is discarded but it contains errors anyway).

Finally, simulations with the S_L soft limit on the number of sur-

vivors lowered to four times the average survivor number showed that practical S_L-limiting neither significantly degraded error-rate performance nor significantly added to computational requirements or buffer backlogs. As long as limit S_L is not too low, the extra computations needed for limiting occur too infrequently to have a significant impact. Along these lines, it is informative to compare soft-limiting to the operation of the M-algorithm. Note that ideal S_L limiting, when invoked, effectively discards all but the best S_L of the paths in contention. Simulations have verified that if this causes the transmitted path to be discarded, the same almost always happens with the M-algorithm using $M = S_L$. Indeed, it is hard to imagine how the M-algorithm, which always keeps a full complement of M lowest metric paths, could avoid finding a set of M paths with metrics lower than the transmitted path when such a set of low-metric paths was found by the T-algorithm causing the loss of the transmitted path after soft-limiting. Soft-limiting therefore is no more harmful than the M-algorithm. The advantage of the proposed T-algorithm over the M-algorithm is the lower average number of paths, and the correspondingly lower average computational load.

VIII. Application

Simulations show that, once lost, the algorithm can have a difficult time recovering the transmitted path, and very long error events are common. This behavior is typical of all reduced-computation (or "reduced-state") algorithms. The frequency and severity of these long events does, however, depend on the threshold and type of trellis structure; long events were not a problem with the ISI responses tested. The simple solution is to periodically force the transmitted path to a known state using ν' known data symbols. This breaks the data into blocks with known start states and provides automatic resynchronization after error. The insertion of these ν' symbols need only cause a very small reduction in effective data rate if they define blocks of length at least, say, $20\nu'$ symbols. With blocks of data, automatic repeat request (ARQ) techniques are possible. Application is envisaged with long (e.g., 1000 symbol) blocks of data containing error-detection check symbols (30 bits of checking will guarantee an undetectable-error rate around 10^{-9}). Note that with data in units of blocks, the lengths of error events inside a block (and the number of symbol errors in the events) are irrelevant.

The choice of the algorithm threshold depends on target block or symbol error rate. If coding is employed, once the threshold is set it seems best to use the longest constraint-length code available. Simulations have verified that this both minimizes state duplications in the survivor list (resulting in a small reduction in the number of survivors) and minimize errors due to Viterbi-like forced decisions at depth L (because of the larger code minimum distance). With such a code selection, there is no basis for contemplating extra algorithm steps for avoiding state duplications.

IX. Conclusion

A reduced-computation breadth-first trellis decoding algorithm has been proposed and analyzed. The decoding effort adapts to the prevailing noise conditions to yield low average effort. This low average computation can be exploited without excessive buffer backlogs and associated decoding delays. Error-rate versus computational complexity behaviour is superior to that of the Viterbi algorithm, and also improves on the M-algorithm. It should be emphasized that the proposed trellis search algorithm is as general as the M-algorithm, and therefore is also a candidate for source coding applications. In addition, the inherent parallelism of the algorithm can be exploited for increased decoding speed in a hardware (possibly VLSI) implementation.

Appendix A

This appendix provides detail on the development of bounds for the error event rate and average number of survivors.

Error Rate Bound

Since the decoding rule will always retain the path of best metric, the transmitted path m can only be rejected if some alternate path, denoted by m', has lower metric. Alternate paths must diverge from the transmitted path m at some point. If this divergence occurs at time $(k-l)\Upsilon$, the path is said to have divergence depth l.

Any alternate path m' that diverges from the transmitted path m with a divergence depth l less than or equal to L is a candidate for best path and therefore a candidate for causing a threshold rejection of path m. Any path m' that has a divergence depth equal to L is a candidate for causing a forced decision rejection of path m at decoding depth L.

An error event is defined by a divergence of the decoder decision path from the transmitted path. Clearly, rejection of the transmitted path is a necessary and sufficient condition for the occurrence of an error event. Assuming that the transmitted path is on the list of contenders, and that it is rejected due to operations at $k\Upsilon$, the resulting error event must begin somewhere in the interval $[(k-L)\Upsilon, k\Upsilon]$ regardless of subsequent decoding operations and resultant changes in the history of the best path.

The total probability of rejecting m at $k\Upsilon$ will be denoted by $P_R(k)$, and the probability of an error event starting at $n\Upsilon$ will be denoted by $P_E(n)$. Now as time moves forward, the L-window inside which an error event may start slides along. If $P_R(k)$ is averaged over an infinite length path, it is then clear that this will yield the same result as averaging $P_E(n)$ over all time; a rigorous argument is given in [14]. It therefore suffices to calculate time-average P_R to find the per-symbol time-average P_E.

By assumption, the processes that produce the channel noise and information symbols are stationary and ergodic. Also, the decoding rule is fixed and has a $P_R(k)$ that depends only on the transmitted path over the last L symbols, and there are always $b^L - 1$ possible alternate paths for $k \geq L$. It follows then that time average rejection probability equals ensemble average rejection probability for $k \geq L$, and that this ensemble average may be evaluated as the average of $P_R(L\Upsilon)$ over all transmit paths m of length L originating in all S possible start states s_0 at $k = 0$.

An upper bound on the probability of an error event starting will be produced first for general time $k\Upsilon$; stationarity will then be invoked to permit attention to be restricted to specific time $L\Upsilon$. The probability of rejecting specific message m at time $k\Upsilon$ will be denoted by $P_R(m, k)$. Rejection of the transmitted path under the proposed decoding algorithm at $k\Upsilon$ can be caused by a minimum metric alternate path m' only if all of the following conditions are met.

1) The transmitted path has not been rejected earlier, that is, it is on the list of contenders at $k\Upsilon$.
2) Alternate path m' has not been rejected earlier, that is, it also is on the list of contenders.
3) Path m' has metric $\Gamma_{m'}$ which is better than the metrics of all other paths in contention at $k\Upsilon$, that is, $\Gamma_B = \Gamma_{m'}$.
4) For threshold rejection a), OR forced decision rejection b):
   a) $\Gamma_m - \Gamma_{m'}$ exceeds threshold T.
   b) m' has divergence depth L and Γ_m exceeds $\Gamma_{m'}$.

Define the signal space regions where received signal vector y may fall such that conditions 1 through 3 above are satisfied for a specific alternate m', and which cause threshold rejection and depth-L forced decison rejection of m, respectively, by $\theta(m, m'_{k-l})$ and $\phi(m, m'_{k-L})$. The subscripts on the m' refer to the time at which m' diverges from transmitted path m, that is, l is the divergence depth.

The regions θ and ϕ as defined above for each m' are all nonoverlapping regions in the signal space. To ease the evaluation, restrictions 1 through 3 above may be removed to create larger overlapping regions defined solely by the conditions of 4) above, and a union bound applied. With this redefinition of regions θ and ϕ, a union bound gives

$$P_R(m, k) \leq \sum_{l=1}^{L} \sum_{m'_{k-l}} P[y \in \theta(m, m'_{k-l})] + \sum_{m'_{k-L}} P[y \in \phi(m, m'_{k-L})]. \tag{A.1}$$

Averaging over all possible messages m gives

$$P_R(k) \le \sum_m P(m) \left[\sum_{l=1}^{L} \sum_{m'_{k-l}} P_t(m'|m) + \sum_{m'_{k-L}} P_f(m'|m) \right] \quad \text{(A.2)}$$

where $P(m)$ is the probability of message m, and P_t and P_f are used to denote threshold-test and forced-decision probabilities, respectively.

Since stationarity ensures that the results are independent of time $k\Upsilon$ for $k \ge L$, it is sufficient to consider the ensemble average result at time $L\Upsilon$ alone. The form for the bound then becomes

$$P_R \le \sum_m P(m) \left[\sum_{\{m'_l, l=1,L\}} P_t(m'|m) + \sum_{m'_L} P_f(m'|m) \right] \quad \text{(A.3)}$$

where all paths m and m' are confined to the interval $[0, LT]$.

Average Number of Survivors

As with rejection probability P_R, the expected number of survivors will, due to stationarity, be independent of time for $k \ge L$, and may be found by ensemble averaging at fixed time $L\Upsilon$ over all paths of length L and over all S possible start states at $k = 0$. The expected number is expressed simply as

$$\bar{S}_T = \sum_{n=1}^{\infty} nP(S_T = n)$$

where S_T is the number of survivors of the rejection rule.

Define Φ_i as the region in the signal space where y may fall such that signal i will survive. Intersections of these Φ_i define regions of multiple survivors. It is straightforward to show that for any number of signal points, including alternates m' and transmitted point m,

$$\bar{S}_T = \sum_i P(y \in \Phi_i) = \sum_m P(m) \sum_i P(y \in \Phi_i | m) \quad \text{(A.4)}$$

where the inherent dependence of y on transmitted path m through x has been recognized.

The survival condition for path i, which is the complement of the rejection condition, is given by $\Gamma_i - \Gamma_B \le T$ where Γ_B is the lowest of the metrics in the particular set of paths that happen to be in contention at time $k\Upsilon$. The regions Φ_i, which are defined by this metric-difference test, are very complicated. It is possible, however, to get a useful bound conditioned on transmitted path m being on the contender list. Since, by definition, $\Gamma_m \ge \Gamma_B$ as long as transmitted path m is in contention, substitution of Γ_m for Γ_B in the metric-difference condition can be used to create the upper bound

$$\bar{S}_T \le \sum_m P(m) \sum_{m'} P(\Gamma_{m'} - \Gamma_m \le T) + 1. \quad \text{(A.5)}$$

A rigorous derivation of this bound can be found in [14].

Appendix B

This appendix contains explicit expressions for the constituent probabilities of the upper bounds of (2) and (4) when applied to the BSC and the AWGN channel. The $\lambda = 1/2$ Chernoff bounds appear on the far right-hand side of the expressions. Note in the following that x and x' are the channel symbol sequences corresponding to transmitted message m and alternate message m'.

AWGN (Additive White Gaussian Noise) Channel

Double-sided noise spectral density is $N_o/2$. Working metrics are squared Euclidian distances with a squared Euclidian distance threshold T_E. The Chernoff bound differs from a standard exponential bound on the Q-function by only a constant factor (both are exponentially tight).

$$P_t(m'|m) = P[\Gamma_m - \Gamma_{m'} > +T] = Q\left(\frac{R(x,x')}{\sigma}\right) < \exp\left(-\frac{T_E}{2N_o}\right) \exp\left(-\frac{d^2(x,x')}{4N_o}\right)$$

$$P_f(m'|m) = P[\Gamma_m - \Gamma_{m'} \ge +0] = Q\left(\frac{d(x,x')/2}{\sigma}\right) < \exp\left(-\frac{d^2(x,x')}{4N_o}\right)$$

$$P_s(m'|m) = P[\Gamma_m - \Gamma_{m'} \ge -T] = Q\left(\frac{d(x,x') - R(x,x')}{\sigma}\right) < \exp\left(+\frac{T_E}{2N_o}\right) \exp\left(-\frac{d^2(x,x')}{4N_o}\right)$$

where $R(x, x') = [T_E + d^2(x, x')]/2d(x, x')$ and $\sigma^2 = N_o/2$.

BSC (Binary Symmetric Channel)

Channel crossover probability is p. Working metrics are Hamming distances used with a Hamming distance threshold T_H. Hamming distance $d_H(x, x')$ has been abbreviated to d_H.

$$P_t(m'|m) = P[\Gamma_m - \Gamma_{m'} > +T] = \sum_{[(d_H+T_H)/2]+1}^{d_H} \binom{d_H}{i} p^i (1-p)^{d_H - i} < \exp\left(-\frac{zT_H}{2}\right) Z^{d_H/2}$$

$$P_f(m'|m) = P[\Gamma_m - \Gamma_{m'} \ge +0] = \sum_{[(d_H+1)/2]}^{d_H} \binom{d_H}{i} p^i (1-p)^{d_H - i} < Z^{d_H/2}$$

$$P_s(m'|m) = P[\Gamma_m - \Gamma_{m'} \ge -T] = \sum_{[(d_H-T_H+1)/2]}^{d_H} \binom{d_H}{i} p^i (1-p)^{d_H - i} < \exp\left(+\frac{zT_H}{2}\right) Z^{d_H/2}$$

where $Z = 2\sqrt{p(1-p)}$ and $z = \ln[(1-p)/p]$.

Acknowledgment

The author wishes to thank the reviewers for suggestions which have improved the clarity of the presentation.

References

[1] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979.

[2] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[3] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A sur-

vey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, pp. 169–176, Feb. 1984.

[4] F. L. Vermeulen, "Low complexity decoders for channels with intersymbol interference," Ph.D. dissertation, Inform. Syst. Lab., Stanford Univ., CA, May 1975.

[5] G. J. Foschini, "A reduced-state variant of maximum likelihood sequence detection attaining optimal performance for high signal to noise ratios," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 605–609, Sept. 1977.

[6] S. J. Simmons and P. H. Wittke, "Low complexity decoders for constant envelope digital modulations," *IEEE Trans. Commun.*, vol. COM-31, pp. 1273–1280, Dec. 1983.

[7] K. R. Matis and J. W. Modestino, "Reduced-search soft-decision trellis decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 349–355, Mar. 1982.

[8] J. L. Massey, "Variable-length codes and the Fano metric," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 196–198, Jan. 1972.

[9] J. K. Omura and M. K. Simon, "Modulation/demodulation techniques for satellite communications: Part IV," Jet Prop. Lab., California Inst. Technol., Pasadena, CA, Nov. 1981.

[10] E. Zehavi and J. K. Wolf, "On the performance evaluation of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 196–202, Mar. 1987.

[11] S. Lin and D. J. Costello, Jr., *Error Control Coding*. Englewood Cliffs, N.J.: Prentice-Hall, 1983.

[12] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[13] J.-E. Porath and T. Aulin, "New results on trellis coding with an extended set of channel symbols," Tech. Rep., Div. Inform. Theory, Chalmers Univ. of Technol., Goteborg, Sweden, June 1985.

[14] S. J. Simmons, "A reduced-computation trellis decoder with inherent parallelism," Ph.D. dissertation, Queen's Univ., Kingston, Ont., July 1986.

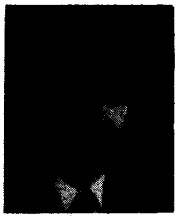
[15] J. B. Anderson, T. Aulin, and C.-E. Sundberg, *Digital Phase Modulation*. New York: Plenum, 1986.

[16] P. G. Gulak and E. Shwedyk, "VLSI structures for Viterbi receivers," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 142–154, Jan. 1986.

[17] S. Mohan and A. K. Sood, "A multi-processor architecture for the (*M*, *L*)-algorithm suitable for VLSI implementation," *IEEE Trans. Commun.*, vol. COM-34, pp. 1218–1224, Dec. 1986.

[18] S. J. Simmons, "A non-sorting VLSI structure for implementing the (M, L) algorithm," *IEEE J. Select. Areas Commun.*, vol. 6, Apr. 1988.

★



Stanley J. Simmons was born in Sudbury, Ont., Canada, on October 9, 1954. He received the B.Sc., M.Sc., and Ph.D. degrees from Queen's University, Kingston, Ont., in 1976, 1981, and 1986 respectively.

From 1976 to 1979 he was employed with Ontario Hydro. From 1981 to 1982 he was with Miller Communications Systems, Kanata, Ont., working on satellite communications and signal processing. He currently holds a University Research Fellowship from the Natural Sciences and Engineering Research Council of Canada (NSERC), and is with the Electrical Engineering Department at Queen's University. His research is concerned with low-complexity high-speed algorithms and architectures for the reception of communications signals.