CASCADED TREE CODES

JOHN L. RAMSEY

TECHNICAL REPORT 478

SEPTEMBER 1, 1970

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
RESEARCH LABORATORY OF ELECTRONICS
CAMBRIDGE, MASSACHUSETTS 02139

The Research Laboratory of Electronics is an interdepartmental laboratory in which faculty members and graduate students from numerous academic departments conduct research.

The research reported in this document was made possible in part by support extended the Massachusetts Institute of Technology, Research Laboratory of Electronics, by the JOINT SERVICES ELECTRONICS PROGRAMS (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract No. DA 28-043-AMC-02536(E), and by the National Aeronautics and Space Administration (Grant NGL 22-009-013).

Requestors having DOD contracts or grants should apply for copies of technical reports to the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314; all others should apply to the Clearinghouse for Federal Scientific and Technical Information, Sills Building, 5285 Port Royal Road, Springfield, Virginia 22151.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Technical Report 478 September 1, 1970

CASCADED TREE CODES

John L. Ramsey

Submitted to the Department of Electrical Engineering at the Massachusetts Institute of Technology in June 1970 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

(Manuscript received July 24, 1970)

Abstract

Cascaded codes are long codes that are constructed by successively encoding a series of relatively short constituent codes. The purpose of cascading is to facilitate decoding by dividing the composite decoding process into a sequence of relatively simple steps, each of which corresponds to the decoding of one of the constituent codes.

In this report cascading techniques in which the constituent codes are tree codes are studied. We determine the efficiency attainable with cascading, and bound the attainable error probability in terms of the composite decoding complexity. Our major results in these areas are the following.

1. A 2-stage cascaded tree code can be formulated to yield an error exponent that equals 1/2 of the single-stage error exponent at all rates below capacity.

2. If N is the composite decoding complexity per decoded symbol for a cascaded tree code in which maximum-likelihood decoding is applied to each constituent code, it is possible to find, in the limit of asymptotically large N, a code for which the decoding error probability becomes arbitrarily close to $(1/N)^{(C/R)}$.

3. It is possible to use sequential decoding on the outer stage of a cascaded tree code and yet communicate at a composite rate exceeding R_{comp}, provided that the alphabet sizes of the constituent codes are suitably restricted.

We also show how to apply the Viterbi decoding algorithm to an unterminated tree code, and describe the burst characteristics of decoding errors made by a Viterbi decoder. Finally, we present techniques for efficiently realizing a useful class of synchronous interleavers.

TABLE OF CONTENTS

I. INTRODUCTION 1

1.1 Communication in the Presence of Noise 1

1.2 Communication over Discrete Memoryless Channels 1

1.3 Cascading of Codes 4

1.4 Outline of the Report 5

II. PROPERTIES OF TREE CODES 7

2.1 Structure 7

2.2 Classification 11

2.2.1 Random Tree Codes 11

2.2.2 Convolutional Codes 12

2.3 Capabilities 15

III. TREE-CODE DECODING TECHNIQUES FOR MEMORYLESS CHANNELS 20

3.1 Sequential Decoding 21

3.1.1 Basic Description 21

3.1.2 Error Probability and Computational Failure 23

3.2 Viterbi Decoding Algorithm 24

3.2.1 Computational Procedure 25

3.2.2 Distribution of the Decoding Lag 30

3.2.3 Character of Error Patterns 38

3.2.4 Decoding Complexity for Systematic Convolutional Codes 41

IV. SIMULATION OF THE VITERBI DECODING ALGORITHM 45

4.1 Distribution of the Decoding Lag 48

4.2 Analysis of Decoding Errors 51

4.2.1 Burst-Error Statistics 53

4.2.2 Error Probability as a Function of the Decoding Lag 53

4.2.3 Comparison with the Coding Theorem 56

4.2.4 Projections 56

V. CASCADED TREE CODES 57

5.1 Review of Block-Code Cascading 57

5.1.1 Product Codes 57

5.1.2 Concatenated Codes 59

CONTENTS

5.2 Formulation of Cascaded Tree Codes 61
5.2.1 Productlike Codes 61
5.2.2 Concatenationlike Codes 63
5.2.3 Burst-Error-Correcting Outer Codes 66
5.3 Efficiency of Two-Stage Cascading 67
5.3.1 Coding Theorem Efficiency 67
5.3.2 Computational Efficiency 73
5.4 Practicality of Cascading 80
5.4.1 Estimated Performance 80
5.4.2 Sequential Decoding at Rates Exceeding R_{comp} 86

VI. REALIZATION OF OPTIMUM INTERLEAVERS 88
6.1 Introductory Concepts 88
6.2 Four Basic Interleaving Techniques 90
6.2.1 Type I (n_2, n_1) Interleaver 90
6.2.2 Type II (n_2, n_1) Interleaver 92
6.2.3 Type III (n_2, n_1) Interleaver 92
6.2.4 Type IV (n_2, n_1) Interleaver 93
6.3 Optimality of Encoding Delay 94
6.4 Reduction and Optimality of Storage 95

Appendix A Proof of Theorems 8 and 9 100

Acknowledgment 105

References 106

I. INTRODUCTION

1.1 COMMUNICATION IN THE PRESENCE OF NOISE

Were it not for noise and distortion, there would be no difficulty in accurately transmitting messages from one point to another. The recipient of an attenuated version of the signal representing a message could amplify the signal with a noiseless amplifier to obtain an exact copy of the transmitted signal. It is therefore the corruption of signals by noise and distortion that engenders the need for information theory and associated signal-processing techniques.

Until the late 1940's communication engineers believed that this corruption limited the accuracy with which transmitted messages could be reproduced by the receiver. They thought that no amount of signal processing could increase the reliability of message reproduction beyond a level that depended on parameters like the signal-to-noise ratio.

Modern communication theory began with Shannon's[1, 2] publication of hitherto astonishing discoveries. He demonstrated that with enough signal processing it is possible to transmit discrete selections over corruptive channels with arbitrarily high reliability, provided only that the rate at which information is conveyed is kept below a value called channel capacity which depends on the corruptive properties of the channel. He did not elaborate, however, on the realization of practical signal-processing techniques, nor on the amount of reliability attainable with a given amount of signal processing. Indeed these are very difficult questions that have occupied the efforts of communication theorists for the last two decades, and which are still largely unsolved. Gallager[3] has presented a summary of many contemporary results.

1.2 COMMUNICATION OVER DISCRETE MEMORYLESS CHANNELS

It is difficult to be precise without considering a specific model for the communication process. One of the simplest models that represents a reasonably large class of noisy physical communication channels is the discrete memoryless channel (DMC), which has a finite input alphabet of K symbols, a finite output alphabet of J symbols, and is characterized by a time-invariant set of channel transition probabilities $\{p_{jk} = \Pr(\text{output} = j \mid \text{input} = k)\}$, where k belongs to the input alphabet, and j to the output alphabet. The effects of noise and distortion are reflected in the values of the channel transition probabilities.

Suppose that a block of N channel symbols is used to represent one of M messages. The parameter

$$R = \frac{1}{N} \ln M, \tag{1}$$

called the information rate, represents the rate at which information is going into the channel. Let the transmitted sequence corresponding to the m^{th} message be

$\underline{x}_m = (x_{1m}, x_{2m}, \ldots, x_{Nm})$, $1 \leq m \leq M$, and let the corresponding received sequence be $\underline{y}_m = (y_{1m}, y_{2m}, \ldots, y_{Nm})$, where the $\{x_{im}\}$ belong to the input alphabet, and the $\{y_{im}\}$ to the output alphabet. Now consider the ensemble of K^{MN} communication systems that are possible by distinguishable assignments of the $\{x_{im}\}$, where the $\{x_{im}\}$ are selected independently from a probability distribution Q(k). Gallager[4,3] has shown that if a maximum-likelihood decoder is used for each communication system, the probability of decoding error averaged over the ensemble of communication systems satisfies

$$\overline{P_{e,m}} = \overline{P_e} < \exp -NE(R), \tag{2}$$

where

$$E(R) = \sup_{0 \leq \rho < 1} \sup_{\{Q(k)\}} [E_o(\rho, Q) - \rho R], \tag{3}$$

and

$$E_o(\rho, Q) = -\ln \sum_{j=1}^{J} \left[\sum_{k=1}^{K} Q(k)\ p_{jk}^{1/1+\rho} \right]^{1+\rho}. \tag{4}$$

The quantity E(R) is called the block code exponent for the channel. Gallager[3] has shown that for a DMC, the E(R) curve looks somewhat like the curve shown in Fig. 1, and has the following properties: There is a rate C called channel capacity for which $E(R) > 0$ for all R, $0 \leq R < C$. There is another rate R_{crit} called the critical rate, $0 \leq R_{crit} \leq C$, for which the E(R) curve of a DMC has a straight-line portion with slope -1 for all R, $0 < R < R_{crit}$. The intercept of the straight-line portion of the E(R) curve with the rate

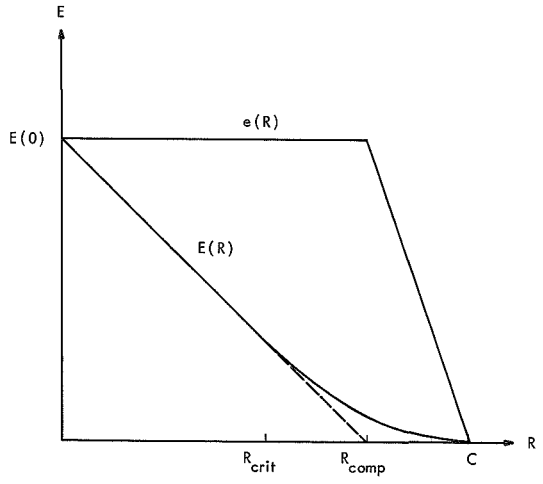

Fig. 1. Block-code and tree-code exponents.

axis is called the computational cutoff rate R_{comp}. It is evident that R_{comp} is numerically equal to E(0), and also that $R_{crit} \leq R_{comp} \leq C$. The rates R_{crit}, R_{comp}, and C all have various engineering significance with respect to communication over the DMC;

some of these properties will be brought out later in appropriate sections of this report.

Each distinguishable assignment of the $\{x_{im}\}$ to represent the messages is called a code, and it represents a particular mapping of the M possible messages into the code words $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_M$. Equation 2 asserts the existence of at least one code for which the probability of decoding error decreases exponentially with the block length N. Shannon, Gallager, and Berlekamp[5] have recently demonstrated that for equally probable messages, there is an exponent $E_L(R)$ such that for all K^{MN} possible codes

$$P_e > \exp -N[E_L(R) - o(N)], \tag{5}$$

where $o(N) \to 0$ as $N \to \infty$. Furthermore,

$$E_L(R) = E(R), \qquad R_{crit} \leq R < C. \tag{6}$$

The preceding discussion suggests that block coding should be an efficient technique for achieving reliable communication over a DMC at all rates below channel capacity. This conclusion follows from (2) and (5), which assert the existence of block codes for which the probability of decoding error decreases exponentially, but no faster, with the block length. Unfortunately, any known decoding technique equivalent to maximum-likelihood decoding of an arbitrary block code requires a decoding effort that grows exponentially with the block length. Thus the probability of decoding error tends to decrease only algebraically with the decoding complexity, thereby substantially reducing the attractiveness of using block coding with maximum-likelihood decoding. Some computationally efficient techniques for decoding particular block codes of particular structures have been discovered, however. Many of these techniques are treated in books by Peterson[6] and by Berlekamp.[7]

Block codes are a subclass of a more general class of codes known as tree codes. The structure and properties of tree codes are treated extensively in Section II. In general, a tree encoder supplies b channel symbols to the channel for each t source symbol that it receives from the source. One of the parameters that characterizes a tree code is its constraint length ν, which can roughly be interpreted as meaning that each set of b channel symbols supplied to the channel depends on, in some sense, only the last νt source symbols supplied by the message source. The constraint length in channel symbols (that is, νb) of a tree code is analogous to the block length of a block code. There are ensembles of tree codes for which the probability of decoding error per source symbol is bounded:

$$\overline{P_e} < \exp -\nu b[e(R) - o(\nu b)], \tag{7}$$

where e(R) is the tree code exponent for the channel, and $e(R) > 0$ for all R, $0 \leq R < C$. Moreover, e(R) is substantially greater than E(R), especially at rates approaching channel capacity. The comparison of e(R) and E(R) is shown in Fig. 1. In Section II it is shown that

$$e(R) = E(0), \qquad 0 \leq R \leq R_{comp}, \tag{8}$$

and e(R) then declines to zero approximately linearly with increasing rate for $R_{comp} \leq R \leq C$.

There are at least two good reasons for studying encoding and decoding techniques for tree codes. One reason is that the tree code exponent is substantially greater than the block code exponent at rates approaching channel capacity, which suggests that tree codes may be much more efficient than block codes in providing reliable communication over a DMC. The second reason is the existence of sequential decoding, which is a simple but powerful decoding technique that is applicable to all tree codes.

1.3 CASCADING OF CODES

Since the code exponents E(R) and e(R) may be small at rates approaching channel capacity, (2) and (7) suggest that the constraint length of an efficient code may have to be large in order to drive the probability of decoding error below some acceptable level. Meanwhile the decoding complexity grows — often very rapidly — with the constraint length. The coding problem is to find good, long codes that can be easily decoded.

One effective general approach to the coding problem is cascading. This technique is illustrated in Fig. 2 for a code with two stages of cascading. The basic idea of cascading is quite simple: A code with a long constraint length is constructed by cascading

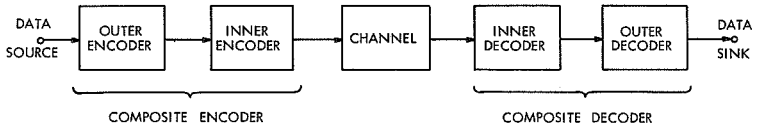

Fig. 2. Two-stage code cascading.

the outputs of two or more constituent encoders, each of which generate a code with a relatively short constraint length. There are several techniques for cascading block codes in which the composite constraint length is equal to the product of the constraint lengths of the constituent codes. Some of these techniques require symbol interleaving between successive stages of coding, in which case the encoders and decoders shown in Fig. 2 are assumed to contain the appropriate interleaving and unscrambling devices. The decoding of cascaded codes is accomplished by successively decoding the constituent codes, stage by stage. Thus the composite decoding complexity is equal to the sum of the decoding complexities of the constituent codes. Cascading is an effective encoding technique whenever it can yield a substantial reduction in the decoding complexity required to attain a given decoding error probability at a given information rate.

Two classes of cascaded block codes that have been extensively studied are Elias's[8] product codes and Forney's[9,10] concatenated codes. These coding techniques will be

reviewed in Section V. Forney[9] showed, however, that it is possible to construct long block codes involving many stages of coding for which the decoding complexity is proportional to N^3, while the probability of decoding error can be made nearly exponential in N:

$$P_e < \exp -K(R)N^{(1-\Delta)}, \tag{9}$$

where Δ is a nonzero, positive quantity that can be made arbitrarily small. Thus in the limit of high decoding complexity and low probability of error, cascading of block codes achieves a probability of error that decreases nearly exponentially with decoding complexity.

Forney[9, 10] also investigated the properties of concatenated block codes which consisted in exactly two stages of coding. He showed that it is possible to construct a two-stage cascaded block code with composite length N and composite rate R such that the probability of decoding error is exponentially bounded:

$$P_e < \exp -NE_C(R), \tag{10}$$

where $E_C(R)$ is the cascaded error exponent, and

$$E(R) \geq E_C(R) > 0, \qquad 0 \leq R < C. \tag{11}$$

He defined the efficiency of two-stage cascading as the ratio of $E_C(R)$ to $E(R)$. The reciprocal of the efficiency indicates roughly how much longer a cascaded code should be to yield the same probability of decoding error as a single-stage code satisfying (2). For one specific example he found that the efficiency was monotonically decreasing with rate, was 0.5 at R = 0, and approximately 0.02 at R = 0.9C. At high rates, therefore, the cascading of block codes appears to produce a substantial reduction in the coding performance when compared with an efficient single-stage code.

Since tree codes are markedly superior in performance to block codes at high rates, and since cascading is an effective, though perhaps inefficient, technique for constructing long block codes, it is natural to wonder whether cascading could usefully be applied to tree codes, and whether the cascading of tree codes would be superior in some ways to the cascading of block codes. We investigate such questions in this report. Some methods for constructing and decoding cascaded tree codes are presented, with emphasis on techniques that appear to require a minimal amount of implementational complexity. The efficiency of cascading tree codes is also studied, and is found to be greatly superior to the efficiency obtained in the cascading of block codes.

1.4 OUTLINE OF THE REPORT

To study cascaded tree codes, a large amount of introductory material is required. In Section II the reader is introduced to the concept, mathematical structure, and error-correcting capabilities of tree codes. Although little of this material is original, the

author believes that its presentation in this report will provide the reader with a much quicker and deeper understanding of tree codes than could be obtained by reading most of the readily available literature on the subject. In Section III two methods of decoding tree codes are considered which appear attractive for decoding the constituent codes of a cascaded tree code. Some of this material, particularly that concerning the Viterbi decoding algorithm, is new. The new results include efficient application of the algorithm to high-rate systematic convolutional codes, application of the algorithm to unterminated tree codes, a study of the asymptotic distribution of the delay required for unique maximum-likelihood decoding of an unterminated tree code, and an analysis of the expected character of decoding error patterns. In Section IV the results of a computer simulation of Viterbi algorithm decoding of short-constraint-length random tree codes are presented. The principal results are observations concerning the decoding lag distribution, evaluation of the error probability as a function of the decoding lag, and a study of the burst characteristics of decoding errors as a function of rate and constraint length. Section V contains the results that will probably be of most interest. Several methods for constructing cascaded tree codes are considered, including techniques analogous to product coding and concatenation for block codes. The asymptotic coding-theorem efficiency of two-stage cascading of tree codes is then investigated, and for concatenationlike codes is shown to be greatly superior to that obtained in the cascading of block codes, especially at rates approaching capacity. Next, we derive an asymptotic bound on the attainable error probability expressed as a function of the composite decoder complexity for cascaded tree codes in which maximum-likelihood decoding is applied at each stage of decoding. Section V concludes with an examination of the question of which decoding techniques appear to be practical for decoding the constituent codes. The emphasis is on finding decoding techniques that provide reasonable improvement in performance with modest decoding complexity. We show conditions for which it is possible to use sequential decoding on the outermost stage and still operate at a composite rate exceeding R_{comp}. Section VI deals with the efficient realization of the synchronous interleavers that are required to construct some of the classes of cascaded tree codes described in Section V.

II. PROPERTIES OF TREE CODES

2.1 STRUCTURE

One should clearly understand the mathematical and geometric structure of tree codes before attempting to study their other properties. This section, which is based largely on Forney's work,[11] furnishes an introduction to the structure of tree codes.

Tree codes are named for the geometric structure usually associated with the encoding process. For many tree codes a trellislike structure might be a more appropriate geometric representation, so that it would be natural to call these codes "trellis codes." It is unlikely that this renaming will ever occur, however, since the term "tree codes" has become firmly established in the literature of coding.

Tree codes are most naturally applied to an unending sequence of symbols to be encoded. For this reason, it is assumed that the data source supplies an indefinitely long ordered sequence of source letters $\ldots, s_{-1}, s_0, s_1, \ldots$, where each source letter comes from an alphabet of size q. The most common example is the binary source, for which q = 2.

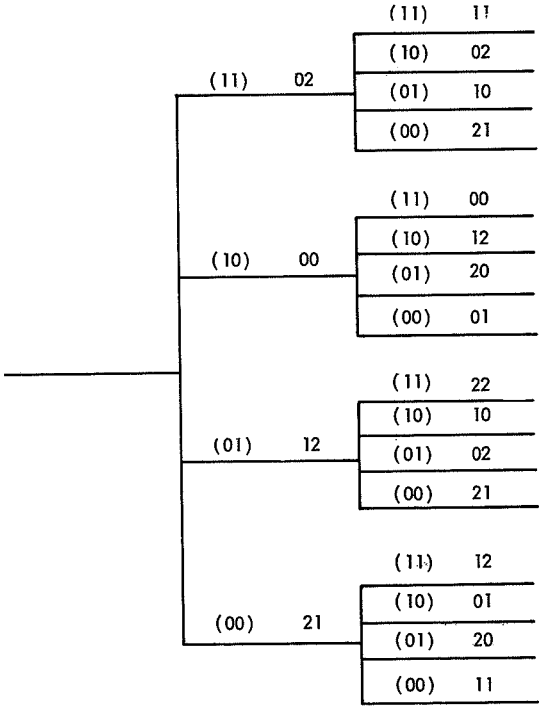

Fig. 3. Example of the encoding process for a tree code.

In a tree encoder the source sequence is partitioned into subblocks containing t contiguous source letters each. Each subblock, containing the t source letters contiguous to the source letters in the adjacent subblocks, is called a source branch, and is encoded into a channel symbol branch comprising a sequence of b channel symbols, where the channel symbols come from an alphabet of size q_c. Figure 3 is an example of a tree encoding process for which q = t = b = 2 and $q_c = 3$. For the tree code shown, the data sequence $\ldots, 1, 0, 1, 0, \ldots$ would be encoded into the channel sequence $\ldots, 0, 0, 1, 2, \ldots$, and the data sequence $\ldots, 0, 1, 1, 1, \ldots$ would be encoded into the channel sequence $\ldots, 1, 2, 2, 2, \ldots$. The tree structure of the encoding process is evident in Fig. 3. A branch of channel symbols is supplied to the channel whenever a source branch is received from the source. The rate of a tree code in nats per channel symbol is thus

$$r = \frac{t}{b} \ln q. \tag{12}$$

One of the most important characteristics of a tree code is its encoding constraint length ν. Precisely, the channel symbols assigned to any branch are determined by only the $\nu + 1$ source branches received immediately before the transmission of the branch. The units of ν are thus source branches, a convention that differs somewhat from the definitions used by other authors. Both the constraint length of a tree code and the block length of a block code are measures of the encoding memories of their respective classes of codes, and both exhibit similar general effects with respect to encoding and decoding complexity and to attainable error probability.

Whenever a tree code has a finite constraint length, there is a rather natural geometric representation of the encoding process that is more trellislike than treelike. (Subsequent discussion throughout this report is restricted to tree codes with finite constraint length.) This representation is easily developed through the concepts of merging and of the state of the encoder.

The concept of the state of the encoder was first defined by Omura.[12] After a channel branch has been specified, the next branch will be determined by only the last ν source branches already received plus the next source branch that will be provided by the source. Accordingly, the state of the encoder can be defined by the ν source branches received immediately before the time at which the most recent channel branch was specified. It is evident that there are $q^{\nu t}$ possible encoder states and that q^t different outcomes can occur for the next channel branch when the encoder is in a particular state, depending on the next t letters received from the data source.

Now consider any two semi-infinite sequences of source symbols S_1 and S_2 supplied after the encoder is in some prespecified initial state. By the definition of constraint length, if the elements of S_1 and S_2 are identical over a sequence of $\nu + 1$ or more contiguous source branches, then the channel symbols selected by the encoder will be identical until the next branch in which S_1 and S_2 differ. S_1 and S_2 are said to be merged at all branches for which the last $\nu + 1$ branches of S_1 and S_2 are identical, and S_1 and S_2 are unmerged at all other branches. Augmenting these definitions, we define a merged span (with respect to S_1 and S_2) as a set of contiguous branches that are all merged, and an unmerged span as a set of contiguous branches that are all unmerged. Corresponding to S_1 and S_2, therefore, there is a sequence of merged spans alternating with unmerged spans. Each span is nonempty and is disjoint from all other spans. The length of a span can be arbitrary, but by definition an unmerged span must contain at least ν branches.

The concepts of merging and of encoder state are illustrated in Fig. 4, which is a trellislike representation of a tree code for which $q = 2$, $t = 1$, and $\nu = 3$. The lightly drawn lines represent the possible branches originating from each encoder state. At each stage of penetration into the trellis there are $q^{(\nu+1)t}$ branches, which correspond to all of the independent combinations of source letters that are possible within the constraint length of the code. Going to and emerging from each state are q^t branches, corresponding to all of the possible combinations of t source symbols

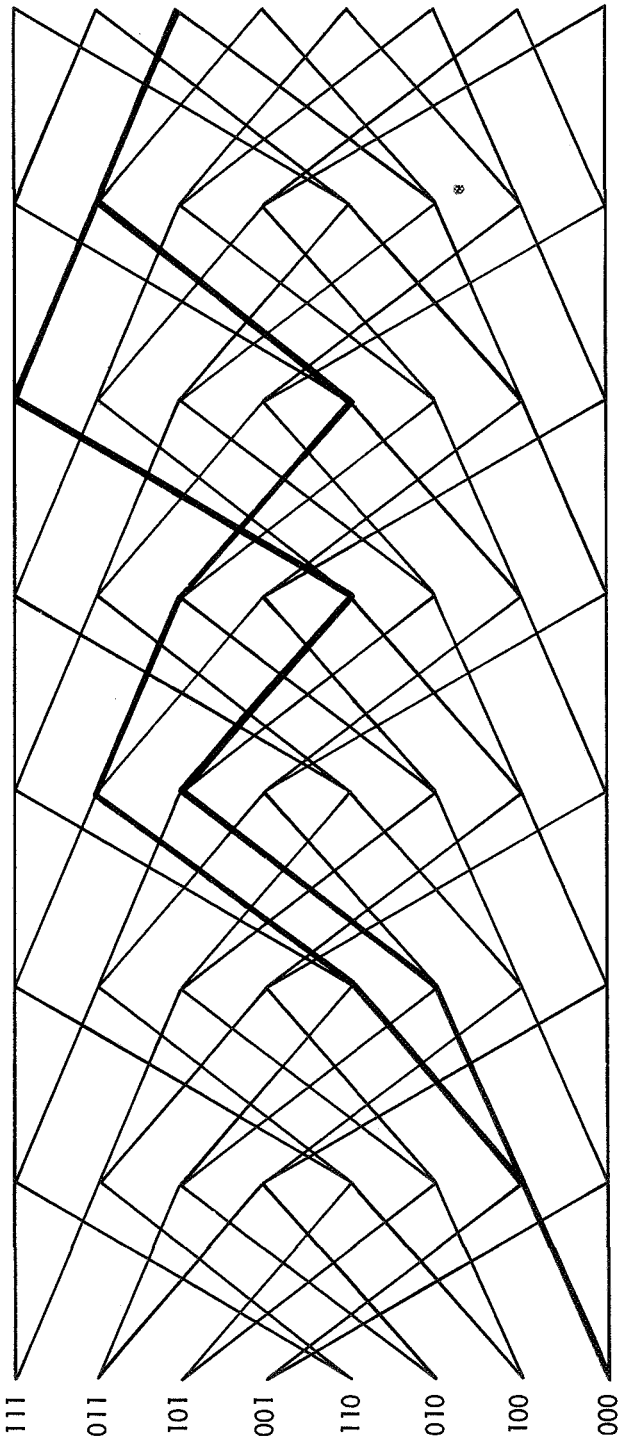

Fig. 4. Trellislike, state-space representation of tree-code encoding process.

occurring in the last source branch and the next source branch, respectively. The set of states having branches going to a particular state are the predecessor states for that state, and the set of states that are immediately reached by the branches emerging from a state are the successor states for that state. Each state has q^t predecessor states and q^t successor states. Observe that there are q^t states that share the same set of predecessor states, and furthermore that these are the only states that can be reached immediately from these q^t predecessor states. One can therefore partition the $q^{\nu t}$ states into subsets in two ways: one partition contains the $q^{(\nu-1)t}$ predecessor sets, and the other partition contains the $q^{(\nu-1)t}$ successor sets. These partitions are generally different.

The set of branches corresponding to a particular sequence of source symbols is called a path. The heavily drawn lines in Fig. 4 are the paths corresponding to the source sequences ...0001101101... and ...0001011101.... These paths clearly show the merging property: The sequences are merged at the initial and final branches shown in the figure, and are unmerged elsewhere. Observe that the length of the unmerged span is ν branches longer than the length of the span in which the source sequences differ.

Figure 4 emphasizes the fact that only a finite number of branch specifications are possible at each stage of penetration into the coding trellis, because of the property that the constraint length is finite. This point is not so easily recognized from the treelike representation shown in Fig. 3, where one might be tempted to infer that the number of branch specifications at any stage of penetration into the code tree grows exponentially with the depth of penetration.

The merging concept is a useful tool for understanding the occurrence of decoding errors in a maximum-likelihood decoder for memoryless channels. Let S_1 be the sequence of symbols supplied by the data source, and suppose that S_2 is identical to S_1, except for a single unmerged span of finite length. The paths corresponding to S_1 and S_2 would then look somewhat like the heavily drawn lines shown in Fig. 4. Furthermore, suppose that the decoder decides that the portion of S_2 included in the unmerged span is more probable than the corresponding portion of S_1. Then a decoding error will certainly occur, since the entire sequence S_2 is more probable than S_1. Of course, there may exist a third sequence, S_3, that is even more probable than S_2 over the entire sequence, and which may be the sequence that is actually decoded. The intersection of the span in which S_1 and S_2 are unmerged with the spans in which S_1 and S_3 are unmerged will be nonempty, however, since otherwise the sequence S_4, which is identical to S_2 over the span in which S_1 and S_2 are unmerged and identical to S_3 elsewhere, is more probable than S_3, thereby contradicting the hypothesis that S_3 is the choice of a maximum-likelihood decoder.

The relationship between merging and maximum-likelihood decoding is used effectively in the Viterbi algorithm for decoding tree codes. This subject will treated extensively in Section III.

It is trivial to observe that tree codes include block codes as a subclass. The simplest example of a class of block codes derived from tree codes is the subclass of tree codes for which $\nu = 0$. The most common subclass of these block codes is the binary codes for which $q = q_c = 2$, $N = b$ is the block length, and the normalized rate is t/N bits per channel symbol. Since tree codes include block codes as a subclass, they must be at least as good as block codes in general.

A much more interesting way of constructing a block code from a tree code is to form a terminated tree code. After every set of K source branches has been supplied by the source, the encoder inserts a resynchronizing sequence of ν fixed, dummy source branches (frequently all "zeros") which is also known to the decoder. After the last dummy source branch, and every $(K+\nu)^{th}$ branch thereafter, the decoder known unambiguously the state of the encoder. Consequently, the successive sequences of $K + \nu$ branches form independent blocks. The tree code is said to be terminated after the $(K+\nu)^{th}$ channel branch, and thus in this way a terminated tree code is made into a block code. The length of the resulting block code is therefore $N = b(K+\nu)$, representing q^{tK} possible code words, so that the block code rate is

$$R = \frac{Kt}{(K+\nu)b} \ln q = \lambda r \tag{13}$$

nats per channel symbol. Forney[11] calls the parameter

$$\lambda = \frac{K}{K + \nu}, \tag{14}$$

the synchronization rate loss.

Wozencraft and Reiffen[13] used terminated tree codes as a practical means of retaining or re-establishing synchronization during the decoding process, while Forney[11] used them analytically to determine bounds on the tree code error exponent in terms of the more easily derived block code error exponent.

2.2 CLASSIFICATION

Several classes of tree codes will be defined here. Each of these classes is useful because it is either analytically tractable or relatively simple to implement in practice. The classifications are based on the properties of the encoding process.

2.2.1 Random Tree Codes

A random tree code is an ensemble of tree codes in which the b channel symbols assigned to each branch are chosen at random independently of each other and are also chosen independently of the channel symbols assigned to all of the other distinct branches in the trellis representation of the code. The channel symbols are selected according to a probability distribution $\{p_k\}$, where p_k is the probability of choosing channel symbol k, $k = 1, 2, \ldots, q_c$. For any specific code in this class,

the channel symbols assigned to a given branch (in time) corresponding to two different source sequences are either identical or totally independent, depending on whether the two sequences are merged at that branch.

It would be absurd to contemplate building an encoder and decoder for a randomly selected tree code, since the equipment complexity would be prohibitive. The usefulness of random tree codes is that they are amenable to relatively simple analysis. We shall exploit that property in section 2.3.

2.2.2 Convolutional Codes

Convolutional codes (sometimes called "recurrent codes") constitute the class of linear tree codes, and these codes are the coding class usually associated with tree codes. Let the source symbols and the channel symbols be elements of the finite field of q elements, GF(q). If X_1 is the channel symbol sequence corresponding to the source sequence S_1, and X_2 is the channel sequence corresponding to S_2, then for a convolutional code the channel sequence $X_1 + cX_2$ will be generated when the source sequence is $S_1 + cS_2$, where c is any element of GF(q). The fact that convolutional codes are linear makes their implementation relatively simple.

We introduce the following notation: Let the t source symbols in the i^{th} branch be designated s_{ij}, j = 1, 2, ..., t, and let the b channel symbols in the i^{th} branch be designated x_{ik}, k = 1, 2, ..., b.. Then for a convolutional encoder, the $\{x_{ik}\}$ can be expressed in terms of the $\{s_{ij}\}$ as follows:

$$x_{ik} = \sum_{\ell=0}^{\nu} \sum_{j=1}^{t} g_{jk\ell}^{(i)}\, s_{(i-\ell)j}. \tag{15}$$

The elements $\{g_{jk\ell}(i)\}$ are elements of GF(q), and they indicate the contribution to x_{ik} made by the source symbol $s_{(i-\ell)j}$. The running variable ℓ in (15) indicates the

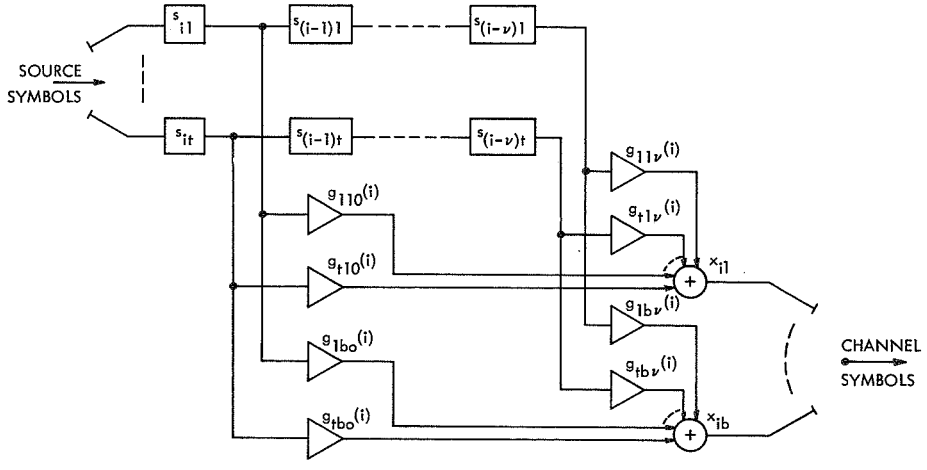

Fig. 5. Generalized realization of a convolutional encoder.

delay in branches since $s_{(i-\ell)j}$ was received; since the constraint length of the code is ν, $g_{jk\ell}^{(i)}$ is identically zero except for $0 \leq \ell \leq \nu$. In terms of i and ℓ, (15) assumes the form of a convolution, which accounts for the term "convolutional code." Figure 5 shows a circuit for realizing (15). This device, adapted from Gallager,[3] comprises two commutators, t $(\nu+1)$-stage shift registers, $(\nu+1)$bt GF(q) multipliers, and b GF(q) adders.

In general, the multipliers $\{g_{jk\ell}^{(i)}\}$ are functions of the branch number i. If, however,

$$g_{jk\ell}^{(i)} = g_{jk\ell}, \qquad \text{all } i, j, k, \ell \tag{16}$$

the code is said to be a time-invariant convolutional code; otherwise, if (16) is not satisfied, the code is a time-variant convolutional code.

The systematic codes are an important subclass of convolutional codes, for which

$$g_{jj0}^{(i)} = 1, \qquad \text{all } 1 \leq j \leq t \tag{17a}$$

$$g_{jj\ell}^{(i)} = 0, \qquad \text{all } 1 \leq j \leq t, \ 1 \leq \ell \leq \nu \tag{17b}$$

$$g_{jk\ell}^{(i)} = 0, \qquad \text{all } 1 \leq j \neq k \leq t, \ 0 \leq \ell \leq \nu. \tag{17c}$$

Equations 17 merely state that the first t symbols in each channel branch are identical to the corresponding source branch symbols. Both time-invariant and time-variant convolutional codes can be systematic. A useful feature of systematic codes is that they can be decoded trivially in the absence of channel errors.

The representation (15) suggests that convolutional codes can be defined in terms of a generator matrix in the same way that linear block codes can be described. Let the source sequence S be represented by the semi-infinite column vector

$$S^T = s_{11}\, s_{12} \cdots s_{1t}\, s_{21}\, s_{22} \cdots s_{2t}\, s_{31} \cdots, \tag{18}$$

and let the channel sequence X be represented by the semi-infinite column vector

$$X^T = x_{11}\, x_{12} \cdots x_{1b}\, x_{21}\, x_{22} \cdots x_{2b}\, x_{31} \cdots. \tag{19}$$

Let G_i be a two-dimensional semi-infinite matrix whose elements are all identically zero except for the submatrix G_i' consisting in rows $[(i-b)+1]$ to ib inclusive and columns $[(i-\nu-1)t+1]$ to it inclusive. The nonzero terms of G_i form the $(\nu+1)t \times b$ array

$$G_i' = \begin{bmatrix} g_{11\nu}^{(i)} \cdots g_{t1\nu}^{(i)} \cdots g_{111}^{(i)} \cdots g_{t11}^{(i)}\, g_{110}^{(i)} \cdots g_{t10}^{(i)} \\ \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ g_{1b\nu}^{(i)} \cdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdots\ g_{tb0}^{(i)} \end{bmatrix} \tag{20}$$

Define G to be the matrix sum of the $\{G_i\}$, subject to the additional condition that $g_{mn} = 0$ for $m \leq 0$ or $n \leq 0$. (The condition is equivalent to the assumption that the encoder is in the all-zero state before the first source branch is supplied by the data source.) Then it follows that

$$X = GS, \tag{21}$$

where G is a two-dimensional semi-infinite matrix that defines the convolutional code specified by (15).

An alternative description for a convolutional code is the parity-check matrix representation given by Wyner and Ash.[14] In this representation, a convolutional code is defined in terms of a two-dimensional semi-infinite parity-check matrix H such that for every possible channel sequence X

$$HX = 0. \tag{22}$$

The parity-check matrix representation is the dual of the generator matrix representation, and for a time-invariant convolutional code with a unique inverse Forney[15] has shown that it is always possible to find a parity-check matrix H corresponding to a given generator matrix G, or vice versa. This is generally a difficult problem, however, except in the case of systematic convolutional codes. In that case, Wyner[16] has shown that the array (20) has the form

$$G'_{si} = \left[\begin{array}{c|c|c|c|c} 0_t & 0_t & \cdots & 0_t & I_t \\ \hline G_{i\nu} & G_{i(\nu-1)} & \cdots & G_{i1} & G_{i0} \end{array}\right], \tag{23}$$

where 0_t is the $t \times t$ zero matrix, I_t is the $t \times t$ identity matrix, and the $\{G_{il}\}$, $i = 0, 1, \ldots, \nu$ are $t \times (b-t)$ matrices whose elements define the systematic convolutional code. Define H_{si} as the two-dimensional infinite matrix whose elements are all identically zero, except for the submatrix H'_{si} consisting in rows $[(i-1)(b-t)+1]$ to $i(b-t)$ inclusive and columns $[(i-\nu-1)b+1]$ to ib inclusive, and whose nonzero terms form the $(\nu+1)b \times (b-t)$ array

$$H'_{si} = \left[-G_{i\nu} \,|\, 0_{b-t} \,|\, -G_{i(\nu-1)} \,|\, 0_{b-t} \,|\, \cdots \,|\, -G_{i1} \,|\, 0_{b-t} \,|\, -G_{i0} \,|\, I_{b-t}\right], \tag{24}$$

where 0_{b-t} is the $(b-t) \times (b-t)$ zero matrix, and I_{b-t} is the $(b-t) \times (b-t)$ identity matrix. Let H_s be the matrix sum of the $\{H_{si}\}$, again subject to the condition that $h_{mn} = 0$ for $m \leq 0$ or $n \leq 0$. Then for the convolutional code defined by (23) and the two-dimensional semi-infinite matrix H_s defined by (24), it follows that

$$H_s X = 0, \tag{25}$$

so that H_s is the parity-check matrix corresponding to the systematic convolutional code whose generator matrix elements are given by (23).

Let the sequence of symbols detected by the receiver be represented by the semi-infinite column vector Y whose entries $\{y_{ik}\}$ are elements of GF(q):

$$Y^T = y_{11}\, y_{12} \cdots y_{1b}\, y_{21}\, y_{22} \cdots y_{2b}\, y_{31} \cdots . \tag{26}$$

Furthermore, suppose that the received sequence can be accurately modeled as the sum of the transmitted sequence and an additive sequence of the channel-introduced errors, where the errors are statistically independent of the transmitted sequence:

$$Y = X + E, \tag{27}$$

where

$$E^T = e_{11}\, e_{12} \cdots e_{1b}\, e_{21}\, e_{22} \cdots e_{2b}\, e_{31} \cdots , \tag{28}$$

and the $\{e_{ik}\}$ are elements of GF(q). As with block codes, the syndrome S_Y corresponding to the received sequence Y is defined as

$$S_Y = HY = H(X+E) = HX + HE = HE, \tag{29}$$

by using (22). Thus the syndrome defines the class of error sequences that was introduced by the channel. One way to realize a maximum-likelihood decoder for additive channels is to determine which member of the class of error sequences defined by the syndrome was most likely to have occurred, and to subtract it from the received sequence to yield the most probable transmitted sequence.

Two important classes of convolutional codes are those for which the normalized rate is either 1/n or (n-1)/n, where n is an integer. Here, these classes are simply called low-rate codes and high-rate codes, respectively.

2.3 CAPABILITIES

We shall conclude with a demonstration that some classes of tree codes are markedly superior to block codes in their ability to correct errors, for a given rate and coding constraint length. The purpose here is to provide motivation rather than rigor. By confining the derivation to a very simple example with limited applicability, we hope to establish the credibility of similar, more general results obtained by others, and to convince the reader that there is a performance advantage to be gained by using certain classes of tree codes. The development here is again based on Forney's work.[11]

Recall from (2) that for random block codes of length N and rate R used on a DMC the average probability that a code word is erroneously decoded after maximum-likelihood decoding is upper-bounded by

$$\overline{P(\epsilon)} < \exp -NE(R), \tag{30}$$

where E(R) is the block code exponent. Gallager[3,4] shows that (30) applies when the probability distribution $\{p_k\}$ of assigning channel input symbols is suitably chosen, and when all symbols in all code words are chosen from that distribution with statistical independence. This condition is then reduced slightly to require only pairwise independence: that is, (30) applies whenever the symbols in any code word are chosen independently of the symbols assigned to any other code word.

A reasonable comparison of block codes and tree codes is a comparison of their error-correcting performance per channel symbol as a function of the encoding constraint lengths. Since (30) is an upper bound on the block error probability, it is also an obvious upper bound on the symbol error probability for a block code. On the other hand, (5) asserts that the block error probability for any code is lower-bounded by

$$P(\epsilon) > \exp -N[E_L(R)+o_1(N)], \tag{31}$$

where $o_1(N)$ is a term that goes to zero with increasing N, and $E_L(R)$ is essentially equal (exactly equal for $R_{crit} \leq R \leq C$) to E(R). Thus an obvious lower bound on the symbol error probability for a block code is

$$\begin{aligned} p_{SB} &> \frac{1}{N} \exp -N[E_L(R)+o_1(N)] \\ &= \exp -N\left[E_L(R)+o_1(N)+\frac{1}{N}\ln N\right] \\ &= \exp -N[E_L(R)+o_2(N)] \end{aligned} \tag{32}$$

and thus, asymptotically, the symbol error probability for a random block code is equal to exp -NE(R) for $R_{crit} \leq R \leq C$.

Now it can be shown that the class of terminated random tree codes is greatly superior to block codes in terms of symbol error probability as a function of encoding constraint length.

It is helpful at first to understand explicitly the geometrical interpretation of a decoding error in a terminated random tree code. For such a code, the encoder state is pre-specified every $K + \nu$ branches, so that all of the paths through the coding lattice converge every $K + \nu$ branches. Define the correct path as the path in the coding lattice that corresponds to the symbols supplied by the data source. If a decoding error occurs in the maximum-likelihood decoding of a terminated tree code, then some other path in the coding lattice must be more probable than the correct path. In particular, the incorrect path must diverge and then remerge with the correct path at least once for a terminated tree code. On the other hand, if there is no path that diverges from and later remerges with the correct path that is more probable than the correct path, then there will be no decoding error for a maximum-likelihood receiver.

The error probability for a terminated random tree code is therefore the probability that no path that diverges from and later remerges with the correct path is more probable than the correct path.

Consider now the set of all paths that diverge from the correct path at branch i and remerge with the correct path after branch i', $1 \leq i \leq K$, $i+\nu \leq i' \leq K+\nu$. Let $M_{ii'}$ be the number of such paths that are distinct. For all of these paths the last ν source branches are identical, so there are only $i' - i - \nu + 1$ source branches in which the symbols may differ. Thus

$$M_{ii'} \leq q^{t(i'-i-\nu+1)}. \tag{33}$$

Conversely, this set of paths includes all of those paths corresponding to source sequences that differ from the actual sequence at the $(i'-\nu)^{th}$ branch and at the $(i+j\nu)^{th}$ branches,

$$0 \leq j < \left\lfloor \frac{i'-i}{\nu} \right\rfloor, \tag{34}$$

where "$\lfloor x \rfloor$" means "the greatest integer contained in x." Thus

$$M_{ii'} \geq q^{t(i'-i-\nu+1)}\, q^{-t\left(\left\lfloor \frac{i'-i}{\nu} \right\rfloor\right)}, \tag{35a}$$

and consequently, using (33), we obtain

$$M_{ii'} = q^{t(i'-i-\nu[1+o(\nu)]+1)}. \tag{35b}$$

For random tree codes, each of the code words corresponding to a divergent path will be independent of the correct word over the span from branch i through branch i'. Since the code words are pairwise independent of the correct word, the union bounding techniques used by Gallager[3,4] apply, so that the set of diverging code words may be regarded as being equivalent to a random block code with $M_{ii'}$ code words and length $N = (i'-i+1)b$. The block code rate is

$$R = \frac{\ln M_{ii'}}{N} = \frac{(i'-i+1-\nu)t \ln q}{(i'-i+1)b} = \mu r, \tag{36}$$

where the tree code rate r was given by (12), and

$$\mu = 1 - \frac{\nu}{i' - i + 1}, \tag{37}$$

so that $0 \leq \mu \leq 1$. Thus the probability of decoding incorrectly over exactly the span from branch i through branch i' is bounded by

$$\overline{P_{ii'}(\epsilon)} < \exp -NE(R)$$

$$= \exp - \frac{\nu b}{1-\mu} E(\mu r)$$

$$= \exp -\nu b e(r, \mu), \tag{38}$$

where the tree code exponent $e(r, \mu)$ is defined as

$$e(r, \mu) = \frac{E(\mu r)}{1-\mu}. \tag{39}$$

Next, define the random tree code exponent $e(r)$ by

$$e(r) = \inf_{\mu : 0 \leq \mu \leq 1} e(r, \mu). \tag{40}$$

Figure 6 provides a graphical construction of $e(r, \mu)$ from $E(R)$, and also compares $e(r)$ and $E(R)$. For $r \leq R_{comp}$, $e(r, \mu)$ is minimized at $\mu = 0$, so that $e(r) = E(0)$, $0 \leq r \leq R_{comp}$.

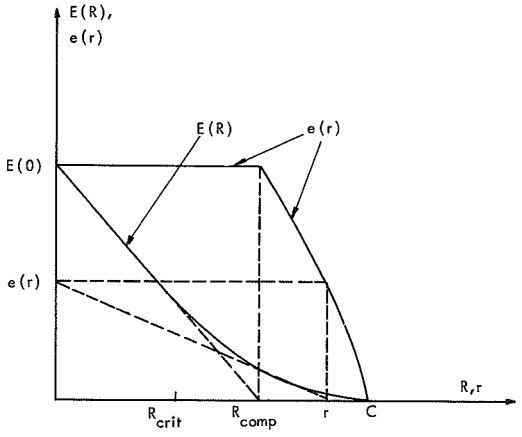

Fig. 6. Construction of e(r) from E(R).

For $R_{comp} < r < C$, $e(r)$ is the E-axis intercept of the line from r on the R-axis that is tangent to the $E(R)$ curve. Thus $e(r) > E(R)$, $0 < r < C$, and the ratio of $e(r)$ to to $E(R)$ increases without bound at rates approaching channel capacity.

Using (40) in (38), therefore, we have

$$\overline{P_{ii'}(\epsilon)} < \exp -\nu b e(r). \tag{41}$$

For the terminated random tree code the union bound yields

$$\overline{P(\epsilon)} \leq \sum_{i=1}^{K} \sum_{i'=i+\nu}^{K+\nu} \overline{P_{ii'}(\epsilon)} < K^2 \exp -\nu b e(r). \tag{42}$$

Asymptotically, therefore, if K does not increase exponentially with ν [for example, by maintaining a fixed synchronization rate loss (14)], then

$$\overline{P(\epsilon)} < \exp -\nu b[e(r)-o(\nu)], \tag{43}$$

and e(r) represents the error exponent for the class of terminated random tree codes.

The derivation leading to (43) applies strictly to a specific class of tree codes with rather limited practical applicability. It does, however, lend credibility to similar results obtained by others. In a similar derivation, Viterbi[17] shows that (43) applies to the class of terminated randomly time-variant convolutional codes. Interestingly, nobody has yet shown that (43) applies to any class of time-invariant convolutional codes; the best error exponent that has been obtained for any of these codes is the block coding exponent E(R). Yudkin[18] and Gallager[3] use a much more elaborate argument based on the properties of sequential decoding to show that (43) applies to the class of unterminated randomly time-variant convolutional codes, where $\overline{P(\epsilon)}$ is then interpreted as the probability that a given branch is decoded incorrectly.

The implications of (43) and Fig. 6 are that, with respect to constraint length, some classes of tree codes are greatly superior to block codes in their ability to correct errors. For a given decoding error probability per symbol, a tree code, whether terminated or not, requires a much shorter encoding constraint length than that required for a block code. The encoding structure inherent in tree codes evidently provides a much more effective use of the stored symbols than that which is obtained in a block encoder. Terminated tree codes using relatively short constraint lengths can be used to form essentially optimum, much longer block codes. We shall later observe that for those block codes that can be realized as terminated tree codes, the tree code realization also yields a substantial reduction in decoding complexity, so that tree codes attain a performance advantage in decoding as well as in reducing error probability.

III. TREE-CODE DECODING TECHNIQUES FOR MEMORYLESS CHANNELS

Although it is relatively easy to find codes that are capable, with maximum-likelihood decoding, of achieving the coding theorem results given by (2) or (7), these codes have not been widely used because the decoding algorithms equivalent to maximum-likelihood decoding for these codes require a computational effort that grows exponentially with the constraint length of the codes. This decoding problem has discouraged the widespread application of coding in operational systems.

One useful but hitherto suboptimum approach to the decoding problem has been to find classes of block codes that have a high degree of mathematical structure which can be successfully exploited to construct easily implemented efficient decoding algorithms. A summary of many of these algebraic coding techniques has been given in books by Peterson[6] and Berlekamp.[7] Unfortunately, none of the known classes of easily decoded algebraic codes contains members that satisfy (2) for arbitrary rates and arbitrarily large block lengths.

The block-code cascading techniques introduced in section 1.3 furnish a second, partially successful, approach to the block-code decoding problem. These techniques have the property that for a nonzero rate, the decoding-error probability can be made arbitrarily small with modest decoding complexity, except for the required symbol storage. None of these techniques quite achieves the coding theorem performance specified by (2), however. Cascading is operationally attractive because it is the only known class of block-code decoding techniques of modest complexity that achieves arbitrary reliability at rates close to capacity.

Several interesting techniques have been presented for decoding tree codes. Of those techniques that are applicable to memoryless channels, three of the most widely studied are sequential decoding, which was introduced by Wozencraft and Reiffen,[13] Massey's[19] threshold decoding, and Viterbi's[17] recent decoding algorithm. Furthermore, there are several useful burst-error-correcting tree-code decoding algorithms, some of which are discussed by Gallager.[3] Of the three memoryless-channel decoding techniques listed above, both sequential decoding and the Viterbi decoding algorithm can be applied to any tree code, and both are capable of attaining the coding theorem performance given by (7). For sequential decoding, however, decoding failures, caused by buffer overflows, dominate the behavior of the decoder, and their probability decreases only as a finite power of the buffer size. On the other hand, the decoding effort for the Viterbi algorithm grows exponentially with the constraint length. Threshold decoding algorithms can be easily implemented, and they do not suffer from the computational problems that characterize the two other decoding methods. Threshold decoding applies only to a limited class of tree codes, however, and it is believed that this class contains no members that satisfy (7) for arbitrary rates and arbitrarily large constraint lengths.

Our purpose is to determine to what extent cascading techniques can usefully be

applied to the construction and decoding of tree codes. In that context, the techniques used to decode the constituent codes of a cascaded tree code must be carefully chosen in order not to render cascading impractical. We shall now examine the properties of those techniques that are reasonable for decoding the constituent codes.

3.1 SEQUENTIAL DECODING

Sequential decoding has been the most extensively studied technique for decoding tree codes, to the point where many communication engineers automatically disregard the possibility of using alternative decoding methods, a conclusion that is understandable although perhaps unwise. Sequential decoding does indeed have some attractive features: It is readily applicable to all classes of tree codes; at rates below R_{comp} the average amount of computation required to decode a branch is small, and its error-correcting capability in the absence of computational failure approaches the bound (7) specified by the coding theorem. On the other hand, the amount of computation required to decode a branch is a random variable c whose frequency of occurrence is upper- and lower-bounded by a finite power of c at all rates. Thus the decoding failure known as a buffer overflow occurs with a probability that decreases only as a finite power of the buffer size. A buffer overflow is also likely to produce a decoding error with substantial error propagation.

3.1.1 Basic Description

The basic principles upon which sequential decoding is based are quite simple. A sequential decoder decodes a tree code by making tentative hypotheses on successive branches and by changing these hypotheses when subsequent choices indicate an earlier incorrect hypothesis. The implementation of a decoder is predicated on two assumptions: (i) the decoder can maintain a replica of the encoder; and (ii) with high probability it can detect an incorrect hypothesis shortly after the incorrect hypothesis has been made. The following simple example, taken from Gallager,[3] illustrates the general application of these ideas in a sequential decoder.

Example 1. Consider the convolutional code generated by the device shown in Fig. 7. The first four branches of the conventional treelike representation of the code, corresponding to Fig. 3, are shown in Fig. 8.

Suppose that the sequence 1100... is supplied by the data source, so that the sequence 111 101 001 000 ..., shown by the heavy line in Fig. 8, is transmitted. We consider two cases.

Case I: No Incorrect Hypotheses

Let the received sequence be 101 101 001 000 .... On the basis of the first received branch, the decoder will tentatively hypothesize that 111 was the first transmitted branch. At this point the received sequence and the hypothesized transmitted sequence differ in one symbol. Continuing, the decoder will hypothesize successively branches 101,

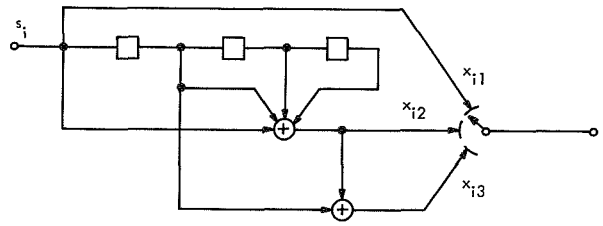

Fig. 7. Encoder for Examples 1 and 2.

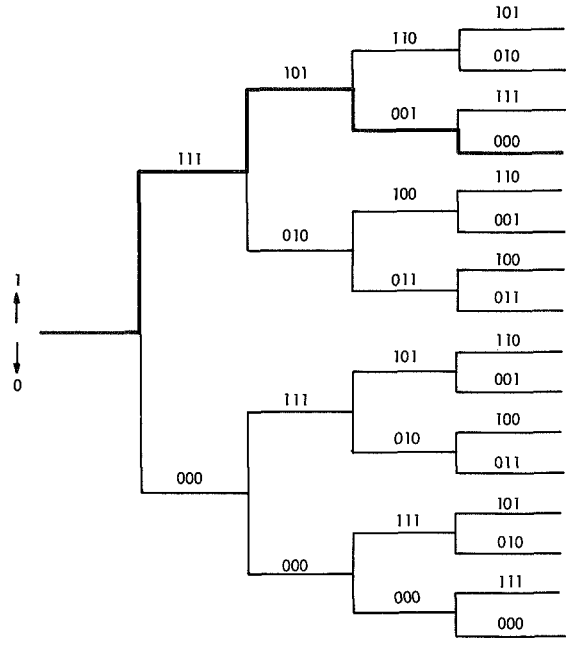

Fig. 8. Treelike representation for the encoder of Fig. 7.

001, and 000, so that after the fourth branch the received sequence and the hypothesized transmitted sequence still differ in only one symbol. The agreement between the two sequences beyond the first branch tends to confirm the validity of the initial hypothesis. Decoding is simplified because in hypothesizing the k^{th} branch the decoder must choose between only two alternatives instead of 2^k alternatives.

Case II: An Initial Incorrect Hypothesis

Suppose instead the received sequence is 010 101 001 000 . . . . This time the decoder will tentatively hypothesize 000 to be the first transmitted branch. Again the received sequence and the hypothesized transmitted sequence differ in only one symbol at this point. Continuing, the decoder will hypothesize successively branches 111, 101, and 001. In this example, therefore, the two sequences differ in k symbols after k branches, k = 1, 2, 3, 4. Once the decoder has made an incorrect hypothesis, its subsequent choices will be between branches that are entirely unrelated to the actual transmitted sequence. The decoder soon recognizes this situation by observing that the disagreement between the received sequence and the hypothesized transmitted sequence grows rapidly with the number of tentatively hypothesized branches. Therefore it must backtrack and make alternative hypotheses in an effort to find a hypothesized transmitted sequence that eventually agrees closely with the received sequence.

Example 1 illustrates the manner in which a sequential decoder hypothesizes successive branches, and the mechanism by which it recognizes that it has somewhere made an incorrect hypothesis. Unfortunately, it gives no indication of the specific mechanics of the procedure to be followed by the decoder once it has detected an incorrect hypothesis. Indeed, the specification and analysis of sequential decoding algorithms is a difficult and complicated problem that has been subjected to considerable research in recent years. It is beyond the scope of this report to reproduce the details of this research here, especially since this subject is ably treated by Gallager,[3] but it is

appropriate to outline here the highlights of this field of research.

3.1.2 Error Probability and Computational Failure

The most extensively studied sequential decoding procedure has been the Fano algorithm, which is a set of rules directing the decoder to hypothesize a subsequent branch, an alternative branch from the current node, or to backtrack, based on the value of a running metric such as accumulated Hamming distance that is monotonically related to the likelihood of the hypothesized transmitted sequence. Using a modification of the Fano algorithm as a model for sequential decoding, Gallager[3] shows that in the absence of computational failure, the decoding error probability per symbol for unterminated randomly time-variant convolutional codes is bounded by (7), which is the coding theorem result for tree codes.

Unfortunately, all sequential decoding algorithms have a property that limits their applicability in practical communication systems. This property, which was studied by Savage,[20] Jacobs and Berlekamp,[21] and Jelenik,[22] is concerned with the peak amount of computation required to decode any branch. The computational effort to decode a branch is a random variable that depends on the number of incorrect hypotheses that are made by the decoder while the branch is within the decoder's memory, and this in turn depends on the channel error sequence. Jelenik[22] has shown that for any $\rho > 0$, the ρ^{th} moment of the computational distribution is bounded by a constant if

$$r < \frac{1}{\rho} E_o(\rho), \tag{44}$$

where

$$E_o(\rho) = \sup_{\{Q(k)\}} E_o(\rho, Q), \tag{45}$$

and $E_o(\rho, Q)$ is given by (4). On the other hand, Jacobs and Berlekamp[21] show that the ρ^{th} moment is unbounded if the sense of the inequality (44) is reversed. [The behavior of the ρ^{th} moment when (44) is satisfied with equality is still unresolved.] In particular, (44) states the well-known result that the average amount of computation required to decode a branch is bounded for $r < R_{comp}$. These results establish that, for rates below capacity, the distribution of decoding computation is Paretian:

$$P(C_o > L) = L^{-\rho(r)}, \tag{46}$$

where

$$\rho(r) = \rho : r = \frac{1}{\rho} E_o(\rho), \tag{47}$$

and thus $\rho(r) > 0$ for $0 < r < C$.

Suppose a sequential decoder has a buffer that can store L branches of received symbols. Then any particular branch must be irrevocably decoded before the next

L branches are received, for otherwise a new branch will be received that cannot be stored, and the decoder will lose sufficient information to continue decoding. This event, calamitous to decoding, is called a buffer overflow. Equation 46 indicates that the probability of a buffer overflow decreases only as a finite power of the buffer size L, while (7) indicates that the decoding-error probability decreases exponentially with the constraint length ν. Thus in the limit of small probability of a decoding failure, the buffer overflow event dominates the erroneous decoding behavior unless the buffer size grows exponentially with the constraint length. This characteristic is common to all sequential decoding algorithms.

A buffer overflow is especially objectionable because it causes the decoder to lose track of the received sequence. Then it is desirable that the decoder become resynchronized quickly, since the decoder is likely to continue producing decoding errors as long as it is unsynchronized. Although tree codes and sequential decoding algorithms do not generally have properties to assist in resynchronization, there are several practical methods that have been used to help re-establish synchronization. For a code with a unique inverse (cf. sec. 3.2.2), a simple non error-correcting decoder can be built to form a reasonable estimate of the correct path. Alternatively, the code can be terminated every few thousand branches, as described in section 2.1. Finally, a feedback channel can sometimes be used to enable the receiver to request the transmitter to repeat its transmission, starting at a mutually known branch in the sequence.

An alternative sequential decoding algorithm has been described by Jelenik[23] and independently by Zigangirov, Pinsker, and Tsybakov.[24] While it offers a higher decoding speed than the Fano algorithm at the expense of increased decoder memory, its attainable error probability and bounds on computational moments are the same as those that are obtained by using the Fano algorithm.

3.2 VITERBI DECODING ALGORITHM

The Viterbi algorithm is a probabilistic decoding procedure that may be readily applied to any tree code. After each branch has been received, the decoder calculates the relative likelihood of each of the $q^{\nu t}$ possible decoder states and the maximum-likelihood path through the coding lattice to each possible state. These calculations are based only on the current received branch and the set of relative likelihoods and maximum-likelihood paths that had been calculated before the branch was received. Strictly, of course, this sort of computation is probabilistically meaningful only for memoryless or Markov channels. Here we assume that the channel is discrete and memoryless. With that assumption, the Viterbi algorithm is a general maximum-likelihood algorithm for decoding codes.

We first describe the computational procedure of the Viterbi decoding algorithm, and then we investigate some of its more important properties.

3.2.1 Computational Procedure

The Viterbi algorithm for decoding tree codes was first formulated by Viterbi,[17] and was later studied by Forney[11] and by Omura.[12] Each of these authors described an algorithm similar to the one presented here, but in which a resynchronizing sequence was periodically inserted into the channel sequence in order to assist the decoder. The formulation described here is stated directly in terms of unterminated tree codes, however, and it indicates clearly when and how much the decoder may decode at any given branch without sacrificing any pertinent information.

The computational procedure described is based on minimum-distance decoding, and is a maximum-likelihood algorithm only for those channels in which the maximum-likelihood path throughout the coding lattice to any possible encoder state is also the path that accumulates the minimum Hamming distance with respect to the received sequence. Such channels are said to be matched to the Hamming metric. Under the assumption of equally likely code sequences $\overline{X}$ for all possible $\overline{X}$, an example of a class of channels that are matched to the Hamming metric is the q-ary symmetric channels, including the binary symmetric channel, with q-ary input and output alphabets whose transition probabilities are given by

$$p(y_j|x_i) = 1 - p, \qquad i = j \tag{48a}$$

$$= \frac{p}{q-1}, \qquad i \neq j. \tag{48b}$$

For these channels the conditional probability of any transmitted sequence $\overline{X}$ of L symbols, given the corresponding received sequence $\overline{Y}$, is given by

$$p(\overline{X}|\overline{Y}) = \frac{p(\overline{Y}|\overline{X})\, p(\overline{X})}{P(\overline{Y})}$$

$$= \frac{p(\overline{X})}{p(\overline{Y})} \prod_{i=1}^{L} p(y_i|x_i)$$

$$= \frac{p(\overline{X})}{p(\overline{Y})} (1-p)^{L-d} \left(\frac{p}{q-1}\right)^{d}, \tag{49}$$

where d is the Hamming distance between $\overline{X}$ and $\overline{Y}$. Since $p(\overline{X})$ is the same for all code sequences $\overline{X}$, $p(\overline{X}|\overline{Y})$ is a monotonically decreasing function of d, provided $(1-p) > p/(q-1)$, so that the q-ary symmetric channels are indeed matched to the Hamming metric.

The reason for considering only a minimum-distance decoding algorithm is that it is relatively simple both to describe and to implement. It is possible to describe a more general algorithm that is maximum-likelihood for any DMC, but this generalized algorithm is much more cumbersome than the minimum-distance algorithm, and the

added complexity would tend to obscure rather than clarify the basic computational procedure.

For minimum-distance decoding, it is easy to describe the iterative process by which a Viterbi decoder calculates the maximum-likelihood path throughout the coding lattice to each of the $q^{\nu t}$ possible encoder states. Before the i^{th} branch is received, let $d_{(i-1)j}$ be the accumulated Hamming distance along the maximum-likelihood, minimum-distance path through the coding lattice to state j, and let $p_{(i-1)j}$ be the sequence of symbols along the maximum-likelihood path to state j, $j = 1, 2, \ldots, q^{\nu t}$. We now show how to calculate d_{ij} and p_{ij} in terms of the i^{th} branch y_i and the $\{d_{(i-1)j}\}$ and the $\{p_{(i-1)j}\}$.

Let SP(j) be the set of predecessor states for state j. Let $x_{ijj'}$ be the set of symbols specified by the encoder during the i^{th} branch from state j' to state j, where $j' \in SP(j)$. Furthermore, let $d_{ijj'}$ be the Hamming distance between $x_{ijj'}$ and the i^{th} branch y_i. After branch i is received, therefore, the Hamming distance that would be accumulated throughout the coding lattice to state j, under the assumption that state j' is the predecessor state for state j, is $d_{(i-1)j'} + d_{ijj'}$. We want to find the minimum-distance path to state j after branch i is received. Define

$$d'_{ij} = \inf_{j' \in SP(j)} [d_{(i-1)j'} + d_{ijj'}], \tag{50}$$

and let state $j'' \in SP(j)$ be any state that achieves this minimization; that is,

$$\{j''\} = \{j'' \in SP(j) : d_{(i-1)j''} + d_{ijj''} = d'_{ij}\}. \tag{51}$$

At this point the decoder may still be unable to specify the maximum-likelihood path p_{ij} with certainty. Consider the set of states $\{j''\}$ defined by (51). Surely $\{j''\}$ is nonempty, since some state must by definition satisfy the condition specified by (51). It is quite possible, however, that $\{j''\}$ contains more than one state; that is, it frequently happens that after branch i is received, there are two or more distinct paths to state j that accumulate the same minimum Hamming distance. We call this condition a decoding ambiguity. Since under our assumptions all of these paths are maximum-likelihood and hence equally likely, the decoder may resolve the ambiguity by arbitrarily selecting a particular state from the $\{j''\}$. Define

$$j''' = \inf_{j'' \in \{j''\}} j''. \tag{52}$$

Then j''' is unique for each i and j. Now the unique path p_{ij} can be specified as the path $p_{(i-1)j'''}$ plus the i^{th} branch from state j''' to state j.

The iterative computational process used by the Viterbi algorithm only requires the relative Hamming distances accrued on the maximum-likelihood paths to each state, and not the absolute distances. To keep the relative accumulated distances within reasonable bounds, especially after decoding indefinitely many branches, we adopt the following

convention. Let

$$d_i = \inf_j d'_{ij}. \tag{53}$$

Obviously $d_i \geq 0$. If $d_i = 0$, there is no problem. If $d_i > 0$, however, the decoder might as well reduce the Hamming distances accumulated along all of the maximum-likelihood paths by d_i, since all of the required quantities are relative rather than absolute. Thus

$$d_{ij} = d'_{ij} - d_i, \qquad \text{all } j, \tag{54}$$

so that by convention

$$\inf_j d_{ij} = 0. \tag{55}$$

Whenever $d_i > 0$, the calculation (54) is called a distance subtraction. The totality of distance subtractions that have been encountered through the L^{th} branch, $\sum_{i=1}^{L} d_i$, is an obvious lower bound on the number of channel errors that have occurred through the L^{th} branch.

For this iterative process the number of computations per branch is a fixed quantity which, unfortunately, grows exponentially with the constraint length of the code.

The preceding formulation described only the mechanics of the iterative computational process, but did not specify how or when the Viterbi algorithm actually decodes source symbols, especially in an unterminated tree code. Now we can be precise about these points. Define the decoding lag k to mean that all $q^{\nu t}$ maximum-likelihood paths $\{p_{ij}\}$ agree everywhere, except for the last k branches. Thus the decoding lag is a random variable that indicates the number of branches about which the decoder has uncertainty as to the unique maximum-likelihood path throughout the coding lattice. From the $(k+1)^{th}$ branch backward, all of the maximum-likelihood paths are identical, so that the decoder may uniquely and unambiguously decode the source symbols up to and including the $(k+1)^{th}$ branch before the current branch. [We emphasize that it is only the decoding that is unambiguous. If the decoded path passes through one or more states having a decoding ambiguity, then of course there would be alternative maximum-likelihood paths throughout the coding lattice. The point here is that the decoded path is a maximum-likelihood path to all states notwithstanding the occurrence of decoding ambiguities.]

The statistical properties of the decoding lag are directly related to the size of the buffer that is required by a Viterbi algorithm decoder to satisfactorily decode unterminated tree codes. If the decoder can store the last L branches of symbols on the maximum-likelihood path to each state, then the probability that $k > L$ clearly bounds the probability of ambiguously decoding a branch if the algorithm is applied to an unterminated tree code. This type of ambiguous decoding is thus caused by a buffer overflow condition. The consequences of a buffer overflow are much less severe for Viterbi algorithm decoding than for sequential decoding, however, because the Viterbi

algorithm has a tendency to resynchronize automatically as it continues its branch-by-branch computations.

The resynchronization property, together with the computational mechanics of the iterative calculations and the decoding procedure, are illustrated by the following example.

Example 2: Consider again the convolutional code generated by the device shown in Fig. 7. Suppose initially that the shift register contains all 0's, and the the data source supplies the sequence 01011001010111... to the encoder. Then the encoder output sequence will be 000 111 010 100 110 001 000 100 010 100 001 100 110 110 .... Suppose the received sequence is 000 111 010 100 100 001 010 100 000 100 001 100 110 100 ... ; that is, there are single transmission errors in the 5^{th}, 7^{th}, 9^{th}, and 14^{th} branches. We illustrate the operation of a Viterbi algorithm decoder under two different assumed conditions.

Case I: The initial state of the encoder, 000, is known to the receiver. Figure 9 illustrates the computational procedures followed by the Viterbi algorithm decoder in this case. At each branch the relative accumulated Hamming distance to each state is shown under that state, subject to (55). For example, at branch 1 only two states are possible — 000 and 100. If 000 is the actual encoder state, its accumulated distance is 0 because the received sequence exactly matches the hypothesized transmitted sequence. Similarly, the accumulated distance to 100 is 3. States marked with a small circle indicate decoding ambiguities where two distinct paths terminating at those states accumulate the same minimum relative distance. For example, the 100 state at branch 4 is ambiguous. Its predecessor states are 000 and 001. The transmitted branch from 000 to 100 is 111, the received sequence is 100, a distance of 2 which when added to the previous distance of the 000 state, 4, yields a total accumulated distance of 6. Similarly, the transmitted branch from 100 to 001 is 100, which exactly matches the received sequence, and adds no Hamming distance. The total accumulated distance along the 001-100 path is therefore just the previous distance of the 001 state, which is also 6. Therefore a decoding ambiguity exists at that state. The decoder, however, arbitrarily keeps only the 000-100 path in accordance with (52). The decoding lag at each branch is also indicated. For example, at branch 4 the decoding lag is 3 because all maximum-likelihood paths to the 8 branch-4 states pass through the initial 000-000 link. Therefore the decoder can then unambiguously decode the first source symbol, which is 0. Similarly, at branch 7 the decoding lag is 4, and the decoder can then decode the next two source symbols, 10. The decoder next decodes a 1 at branch 8, and then waits until branch 14 when it decodes the next 7 symbols, 1001010. The decoded path is shown by the heavy line throughout the coding lattice. Since it passes through no decoding ambiguities thus far, the maximum-likelihood path decoded is in fact unique at this point.

Case II: The initial state of the encoder is not known to the receiver. Figure 10 illustrates the computational procedures followed for the first 9 branches by the Viterbi

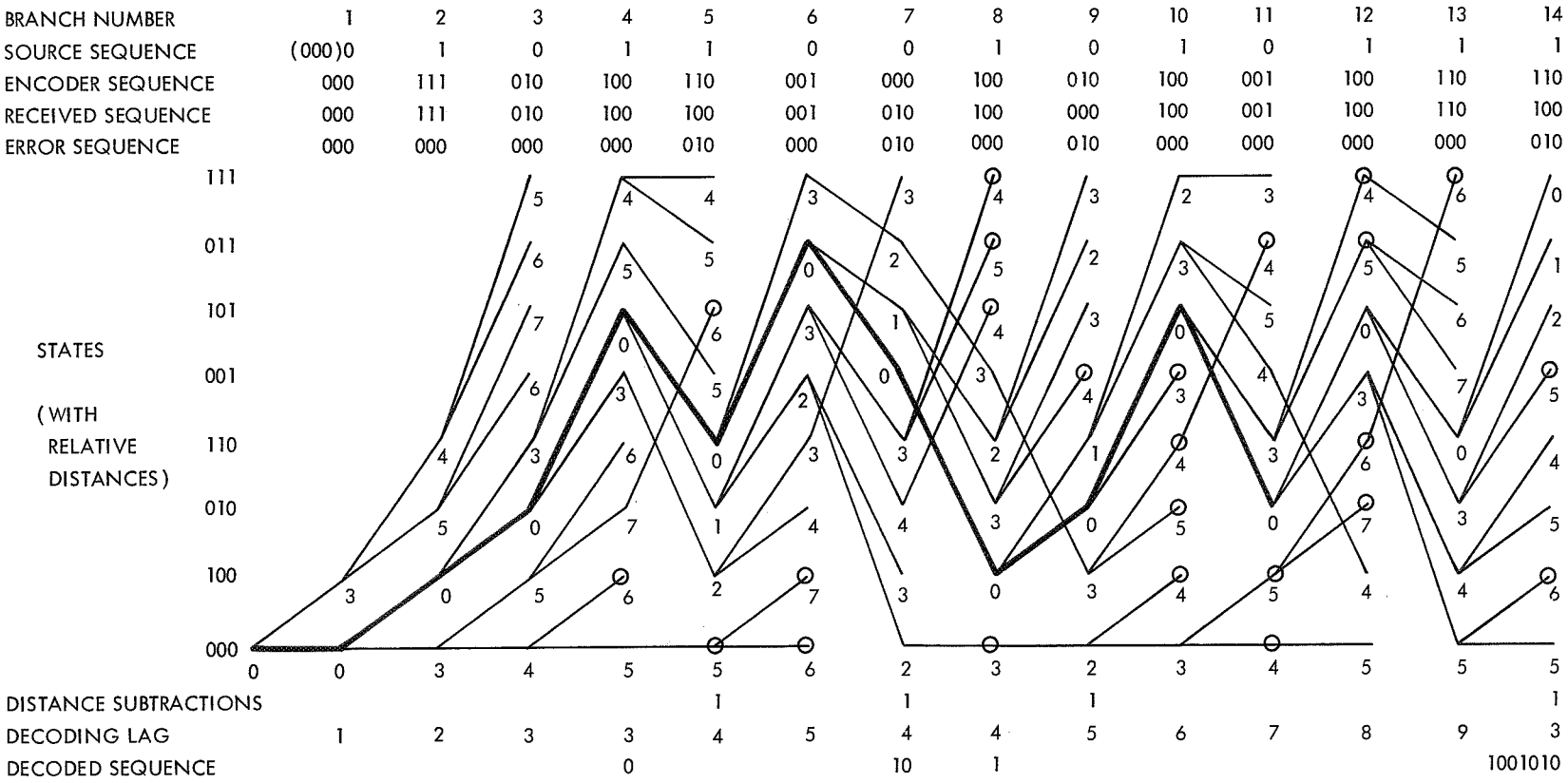

Fig. 9. Viterbi decoding sequence for Example 2, Case I.

algorithm decoder in this case. Since the initial encoder state is unknown, the decoder assumes initially that all states are equally likely, and it reflects this fact by assigning 0 as the relative accrued distance to each state. The decoding and computations then proceed in the obvious, branch-by-branch manner. Observe that at branch 7 the distribution of relative accrued distances is identical to that for Case I, Fig. 9. There is an ambiguity at the 010 state, however, and application of (52) yields 100 as the predecessor state instead of 101. Since the distribution of relative accrued distances is identical for both cases at branch 7, the maximum-likelihood paths beyond branch 7 will therefore be identical for the two cases. At branch 9 the maximum-likelihood paths to all states coincide for both cases, and the decoder is able to decode the first 4 (or 7) source symbols, (000)0101. From this branch on decoding will be identical for both cases.

Example 2 illustrates the tendency of a Viterbi algorithm decoder to find a unique maximum-likelihood path through the coding lattice, regardless of the assumed initial conditions. This property can also be interpreted in another useful way. Whenever a buffer overflow occurs, that is, whenever $k > L$, then the decoder is prone to make decoding errors because it must choose between two or more alternative sets of maximum-likelihood paths. Of course, it should be possible to specify an algorithm for making this choice (such as choosing the source symbol corresponding to the most probable path, or the source symbol that occurs most often at the beginning of all the $q^{\nu t}$ maximum-likelihood paths) that would substantially reduce the error probability, even under these circumstances. Regardless of decoding errors, however, an implicit initial distribution of accumulated relative distances appears in the current set of accumulated relative distances. Example 2 suggests that regardless of this initial distribution, the decoder will eventually find a unique maximum-likelihood path through the coding lattice. Thus the Viterbi algorithm automatically tends to resynchronize after a decoding error or a buffer overflow, in contrast to sequential decoding.

3.2.2 Distribution of the Decoding Lag

The storage requirements and buffer-overflow probabilities of a Viterbi decoder depend on the statistical properties of the decoding lag. We shall demonstrate that for a wide class of applications, the asymptotic distribution of the decoding lag is exponentially bounded; that is, there is a $\beta > 0$ such that, for sufficiently large L,

$$p(k>L) < e^{-\beta L}. \tag{56}$$

The class of applications for which (56) is satisfied includes the minimum-distance decoding of any time-invariant convolutional code with a unique inverse, or to the ensembles of time-variant convolutional codes or random tree codes, operating over a wide class of DMC's, including the reachable channels, and operating at any rate above or below capacity. [A DMC is <u>reachable</u> if all possible transition probabilities $\{p_{ij}\}$ are strictly nonzero. The reachable channels include the q-ary symmetric channels defined by (48). Equation 56 applies for the indicated class of applications, even though

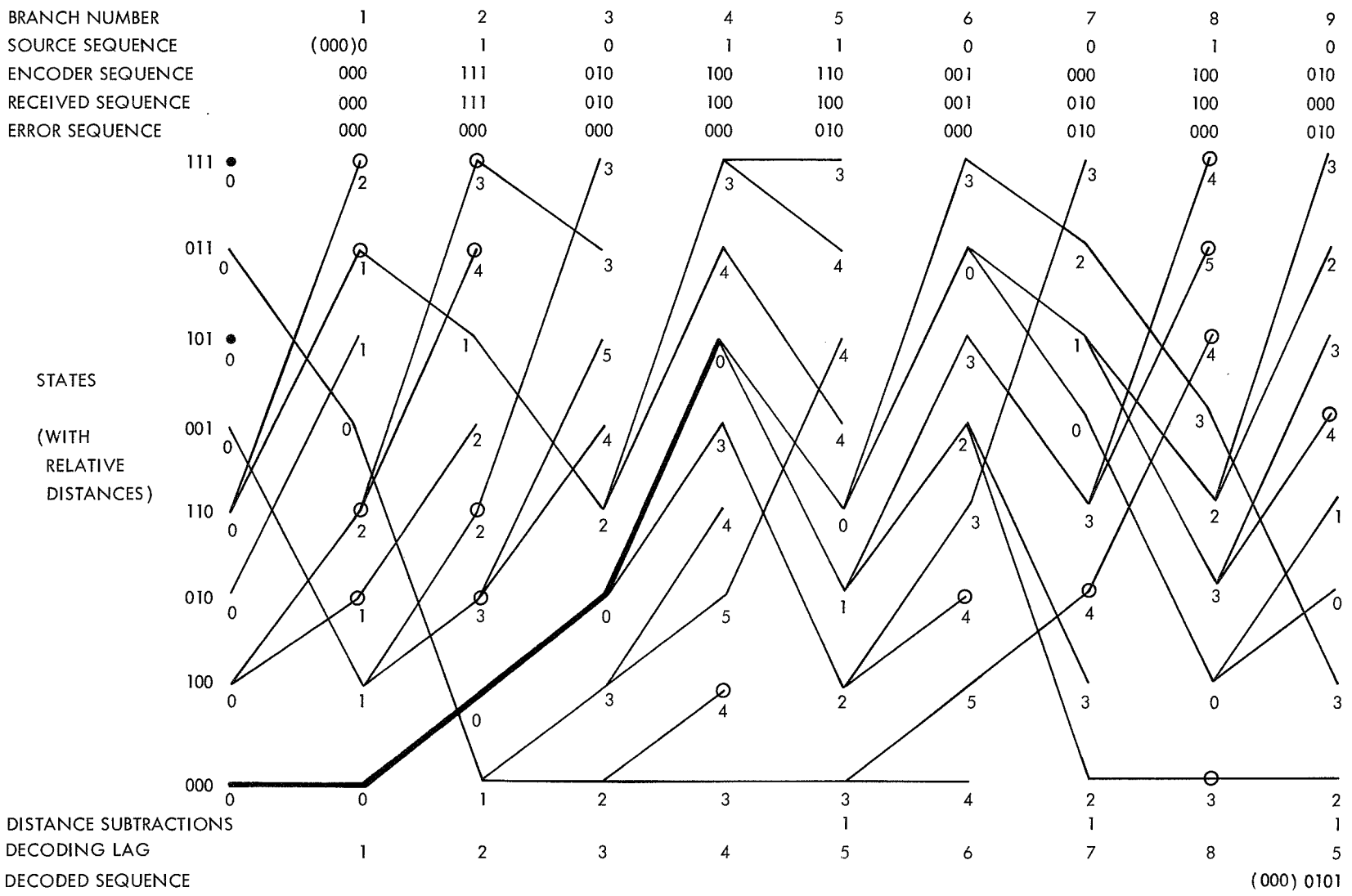

Fig. 10. Viterbi decoding sequence for Example 2, Case II.

minimum-distance decoding is not necessarily maximum-likelihood decoding for an arbitrary reachable channel.]

The essence of the derivation establishing (56) is to model the Viterbi algorithm as a homogeneous Markov process with a finite number of states. Exactly one of these states, which represents a reduction in the decoding lag, is absorbing. We then consider the L^{th}-order transition matrix in the asymptotic case as $L \to \infty$ and show that the probability that the absorbing state has not been entered after L transitions is lower-bounded by $1-\lambda^L$ for some λ such that $0 < \lambda < 1$. This in turn establishes (56).

Theorem 1 is used to show that the Markov process that represents the Viterbi algorithm has a finite number of states.

Theorem 1

$$\sup_j d_{ij} \leq b\nu. \tag{57}$$

Proof of Theorem 1: By (55), $\inf_j d_{(i-\nu)j} = 0$. Let j' be the state that achieves this minimization; that is, $d_{(i-\nu)j'} = 0$. Observe that there is a unique path with exactly ν branches from state j' to any state j. With respect to the received sequence, this path can accumulate an absolute Hamming distance no greater than $b\nu$. Clearly, the minimum absolute Hamming distance accumulated to any state at branch i cannot be negative, and the minimum-distance path to state j must be at least as good as the path from state j'. This establishes (57).

Let the problem of calculating the distribution of the decoding lag be restated in the following manner. Consider the set of maximum-likelihood paths to the $q^{\nu t}$ encoder states at the i^{th} branch, where i is arbitrary. Next consider the set of maximum-likelihood paths to the $q^{\nu t}$ states at the $(i+L)^{th}$ branch, $L > 0$. The probability that the decoding lag at the $(i+L)^{th}$ branch is less than or equal to L is the probability that all of the $q^{\nu t}$ maximum-likelihood paths at the $(i+L)^{th}$ branch pass through a single encoder state at the i^{th} branch.

For each possible encoder state j at the $(i+L)^{th}$ branch, $j = 1, 2, \ldots, q^{\nu t}$, consider the 2-tuple $(d_{(i+L)j}, S_{(i+L)j})$. The quantity $d_{(i+L)j}$ is the relative distance accrued on the maximum-likelihood path to state j at branch i+L, as defined by (54), while $S_{(i+L)j}$ is defined as the encoder state at branch i through which the maximum-likelihood path to state j at branch i+L passes. Since $d_{(i+L)j}$ can assume no more than $\nu b+1$ values and $S_{(i+L)j}$ can assume no more than $q^{\nu t}$ values, the 2-tuple can be specified in no more than $q^{\nu t}(\nu b+1)$ ways.

The Markov state of the decoding process at branch i+L can be specified by the ordered set of $q^{\nu t}$ 2-tuples $\{(d_{(i+L)j}, S_{(i+L)j})\}$, plus the actual state of the encoder at branch i+L. The number N_m of possible Markov states is thus finite and bounded:

$$N_m \leq q^{\nu t}[q^{\nu t}(\nu b+1)]^{q^{\nu t}}. \tag{58}$$

Of course, many of the possible states included in (58) are nonexistent or void; for example, all of the states for which (55) is not satisfied are not allowable under our formulation. Moreover, many states are equivalent because they indicate a reduction in the decoding lag. These are the states for which

$$S_{(i+L)j} = S_{(i+L)1}, \qquad j = 2, 3, \ldots, q^{\nu t}. \tag{59}$$

These equivalent states can all be represented by a single lag-reducing state, denoted z. With respect to the Markov process this state is an absorbing state, since once it has been entered — that is, once the decoding lag has been reduced — the Markov process remains in that state for all subsequent branches.

We now show that for time-invariant convolutional codes with a unique inverse, and for the ensembles of time-variant convolutional codes and random tree codes, there are no other sets of absorbing states in the Markov process representation, so that all of the allowable Markov states except z are transient. To prove this assertion, it suffices to show that it is always possible for the process to reach state z from any allowable state after a finite number of transitions.

First consider a time-invariant convolutional code with a unique inverse. By unique inverse, we mean that the code is not subject to indefinite error propagation, or equivalently that no input sequence containing an infinite number of nonzero symbols can produce an output sequence containing only a finite number of nonzero symbols. Massey and Sain[25] derive the conditions for which a time-invariant convolutional code has a unique inverse. For such a code they show that it is possible to build a linear, feedback-free decoder that uses only the last M received branches to uniquely decode the source sequence, where M is finite. Although this decoder is not generally a maximum-likelihood decoder, it has the property that will always be decoding along the correct path after M or more error-free branches are received.

Theorem 2

For any time-invariant convolutional code with a unique inverse (TICCUI), it is possible to find parameters K_o and $K < K_o$ such that a Viterbi decoder is decoding along the correct path with a decoding lag not exceeding K whenever K_o or more consecutive error-free branches are received.

Some intermediate results must be established before Theorem 2 can be proved.

Lemma 1

For a TICCUI, any source sequence of $M+\nu$ branches that is nowhere merged with the all-zero sequence produces an encoder sequence whose corresponding $M+\nu$ branches are not all zeros.

Proof: Suppose the encoder produces $M+\nu$ all-zero channel branches. Since the Massey-Sain decoder has a memory of M branches, it estimates zeros for the source symbols from the M^{th} through the $(M+\nu)^{th}$ branches. Furthermore, all-zero branches are

the only source branches that could have produced the last $\nu+1$ all-zero channel branches, since the code has a unique inverse that can be recovered by a Massey-Sain decoder. Thus $M+\nu$ consecutive all-zero channel branches implies that the source sequence is merged somewhere with the all-zero sequence, and therefore the lemma must be true.

For notational purposes, define d_{si} as the relative Hamming distance and d'_{si} as the absolute Hamming distance accrued from branch 0 by a minimum-distance Viterbi decoder to state s at branch i. The corresponding distances accrued to the all-zero state will be designated d_{0i} and d'_{0i}.

Lemma 2

Let the source sequence be the all-zero sequence. Let all branches after branch 0 be received without errors. Then for a TICCUI, $d_{0i} = 0$ for $i \geq \nu b(M+\nu)$.

Proof: From (55) and (57), $0 \leq d_{00} \leq \nu b$. Certainly $d'_{0(M+\nu)} \leq d_{00}$, since the branches are received without errors. Now consider $d'_{s(M+\nu)}$, $s \neq 0$. By Lemma 1, if the minimum-distance path to s does not merge anywhere with the all-zero path, then $d'_{s(M+\nu)} \geq 1$. Otherwise $d'_{s(M+\nu)} \geq d'_{0(M+\nu)}$. Suppose $d_{00} > 0$, since Lemma 2 is trivially true otherwise. Then if $d'_{0(M+\nu)} \geq 1$, we see that $d'_{s(M+\nu)} \geq 1$ for all s, and a distance subtraction will occur so that $d_{0(M+\nu)} \leq d_{00} - 1$. Alternatively, if $d'_{0(M+\nu)} = 0$, Lemma 2 is trivially true. Applying this result no more than νb times establishes Lemma 2.

Lemma 3

Assume the same hypotheses as for Lemma 2, and let $d_{00} = 0$. Then after $(\nu b+1)(M+\nu)$ branches, each path in a minimum-distance Viterbi decoder is merged somewhere with the all-zero path.

Proof: Certainly $d'_{0i} = d_{0i} = 0$. Apply Lemma 1 $\nu b+1$ times, then no path corresponding to an unmerged sequence of $(\nu b+1)(M+\nu)$ branches can accumulate an absolute Hamming distance of less than $\nu b+1$. Since, from (57), $d_{si} \leq \nu b$ for all s, Lemma 3 is established.

Proof of Theorem 2: Since the code is a convolutional code, we lose no generality by assuming that the source sequence is the all-zero sequence. Let all branches after branch 0 be received without errors. From Lemma 2, $d_{0i} = 0$, and there are no distance subtractions for $i \geq \nu b(M+\nu)$. Suppose that the minimum-distance path to some state s is merged with the all-zero path at some branch I, where $I \geq (\nu b+1)(M+\nu)$. Then the minimum-distance path to s must be coincident with the all-zero path for all branches i in the range $(\nu b+1)(M+\nu) \leq i \leq I$. If this last statement were not true, then either the minimum-distance path would, looking backward, diverge from the all-zero path at some branch i_2 in the range $(\nu b+1)(M+\nu) \leq i_2 \leq I$ and later remerge with the all-zero path at some branch i_1 in the range $\nu b(M+\nu) \leq i_1 \leq i_2 - \nu$, or it would diverge at branch i_2 but not remerge for any $i_1 \geq \nu b(M+\nu)$. The first case is prohibited by the fact that the code has a unique inverse, so that any such divergent sequence must

produce an output with a Hamming distance of at least one in order not to be confused with the all-zero sequence. The second case is prohibited by Lemma 1 and by the fact that $d_{0[\nu b(M+\nu)]} = 0$. By applying these considerations and Lemma 3, we observe that Theorem 2 is satisfied for $K_o = 2(\nu b+1)(M+\nu)$ and $K = \frac{1}{2} K_o$.

Theorem 2 states that a time-invariant convolutional code with a unique inverse will always reach state z whenever K_o consecutive error-free branches are received. This result implies that the ensembles of time-variant convolutional codes and random tree codes can (with perhaps an extremely small, but nonzero probability) reach state z after K_o branches, provided that the ensemble chooses the same branch specifications as for the time-invariant convolutional code considered above for the next K_o branches, and that these branches are received without errors.

To simplify the derivation of (56), it is convenient to assume that the Markov process is also homogeneous; that is, the stochastic matrix T_i that contains the Markov state transition probabilities from the i^{th} branch to the $(i+1)^{th}$ branch is independent of i. Certainly, the Markov state at the $(i+1)^{th}$ branch is uniquely determined by the Markov state at the i^{th} branch, the i^{th} source branch, and the channel error sequence on the i^{th} branch. For a DMC the channel-error sequence is independent of i, so that the Markov process is homogeneous if the probability assignment of source branches supplied to the encoder is also independent of i. One way to ensure this source independence is to add a pseudo-random sequence to the source output, and later to subtract this sequence from the output of the decoder.

Thus for the class of applications described after (56), the operation of the Viterbi decoding algorithm can be modeled as a homogeneous Markov process with exactly one absorbing state that represents a reduction in the decoding lag. Let T be the finite matrix of one-step transition probabilities characterizing the Markov process, and let T_L be the matrix of transition probabilities after L branches. Since the process is homogeneous,

$$T_L = T^L. \tag{60}$$

Furthermore T can be partitioned:

$$T = \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix}, \tag{61}$$

where A is the $(n-1) \times (n-1)$ matrix whose elements represent the one-step transition probabilities among the transient states, B is the $1 \times (n-1)$ matrix whose elements are the one-step transition probabilities from the transient states to the absorbing state, and the 0 and 1 reflect the fact that z is an absorbing state. It follows that

$$T_L = \begin{bmatrix} A^L & B_L \\ 0 & 1 \end{bmatrix}. \tag{62}$$

Accordingly, we investigate the behavior of A^L. We shall require some results from the Perron-Frobenius theorem.[26]

Perron-Frobenius Theorem: (Condensation)

Suppose A is an irreducible square matrix with real, non-negative elements. Then A has a real positive eigenvalue λ_1 with the following properties:

(i) if a is any other eigenvalue of A, then

$$|a| \leq \lambda_1. \tag{63}$$

(ii) λ_1 increases when any element of A increases.

(iii) $$\lambda_1 \leq \max_j \left(\sum_k a_{jk} \right). \tag{64}$$

These results are accepted without proof here. From property (iii), λ_1 is bounded by the maximum row sum of A. Since A is derived from a stochastic matrix,

$$\lambda_1 \leq 1. \tag{65}$$

Furthermore, A has some row whose row sum is strictly less than 1, since B is assumed to have at least one nonzero element. Applying property (ii), we observe that $\lambda_1 \neq 1$ because if $\lambda_1 = 1$, an element in the row whose row sum is strictly less than 1 could be increased so that $\lambda_1' > 1$, which would finally contradict property (iii). By using property (i), the following lemma is established.

Lemma 4

Let A be the matrix of transition probabilities among the transient states of any Markov process having sets of absorbing states. If λ is any eigenvalue of A, then

$$|\lambda| < 1. \tag{66}$$

Next, observe that the matrix A can be expressed in its Jordan canonical form; that is, if A is an $n \times n$ matrix, there exists a nonsingular $n \times n$ matrix Q such that

$$A = Q \begin{bmatrix} J_{k_1}(\lambda_1) & & \cdots & 0 \\ 0 & J_{k_2}(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & J_{k_m}(\lambda_m) \end{bmatrix} Q^{-1}, \tag{67}$$

where the $\{J_{k_i}(\lambda_i)\}$ are the $k_i \times k_i$ matrices of the form

$$J_k(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & 0 & \dots & 0 \\ 0 & \lambda & 1 & 0 & \dots & 0 \\ 0 & 0 & \lambda & 1 & \dots & 0 \\ . & . & . & . & & . \\ . & . & . & . & & . \\ . & . & . & . & & . \\ 0 & 0 & 0 & 0 & \dots & \lambda \end{bmatrix}, \tag{68}$$

and

$$k_1 + k_2 + \dots + k_m = n. \tag{69}$$

The $\{\lambda_i\}$ are eigenvalues of A, not necessarily distinct, and to every distinct eigenvalue of A there corresponds at least one $J_{k_i}(\lambda_i)$ in (67). Thus

$$A^L = Q \begin{bmatrix} J^L_{k_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & J^L_{k_2}(\lambda_2) & \dots & 0 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & \dots & J^L_{k_m}(\lambda_m) \end{bmatrix} Q^{-1}, \tag{70}$$

where the $h\ell^{th}$ term of $J^L_k(\lambda)$ is

$$j^{(L)}_{k_{(h\ell)}} = \binom{L}{\ell-h} \lambda^{L-\ell-h}, \qquad 0 \leq \ell - h \leq L. \tag{71}$$

Now suppose that A is the matrix of transition probabilities among the transient states of a homogeneous Markov process with a single absorbing state, as in (61). Let λ_A be any eigenvalue of A, and define

$$\lambda^* = \sup |\lambda_A|. \tag{72}$$

From Lemma 4, $\lambda^* < 1$, so that

$$\lambda = \tfrac{1}{2}(1+\lambda^*) < 1. \tag{73}$$

By choosing

$$\lambda_1 = \tfrac{1}{2}(\lambda+\lambda^*), \tag{74a}$$

and

$$\lambda_2 = \frac{1}{2}(\lambda+\lambda_1), \tag{74b}$$

it can be seen that, for sufficiently large L,

$$\left| J^{(L)}_{k_{(h\ell)}} \right| < \lambda_1^L, \tag{75}$$

so that

$$\left| a^{(L)}_{h\ell} \right| < \lambda_2^L, \tag{76}$$

where $a^{(L)}_{h\ell}$ is the $h\ell^{th}$ element of A^L. Thus, in (62),

$$b^{(L)}_j > 1 - n\lambda_2^L > 1 - \lambda^L, \tag{77}$$

where $b^{(L)}_j$ is the j^{th} element of B_L in (62), $1 \leq j \leq n-1$, provided L is sufficiently large. This establishes the following theorem.

Theorem 3

Consider any homogeneous Markov process with a single absorbing state. For sufficiently large L there exists a positive number $\lambda < 1$ such that the probability that the process is not in the absorbing state after L transitions is less than λ^L.

Theorem 3 establishes (56) for the class of applications described after the statement of (56). Not only does (56) apply to individual time-invariant convolutional codes with a unique inverse, but it can also be interpreted to apply to individual codes that belong to the ensembles of time-variant convolutional codes or random tree codes in which the generator connections or branch specifications are selected randomly at each branch. While (56) does not apply to all possible codes that can be picked from these ensembles, it can be interpreted to mean that the probability of selecting an individual code for which the decoding lag does not converge is an exponentially decreasing function of the number of branches that have been selected.

Although the preceding derivation establishes the validity of (56), it gives little insight into a reasonable estimate of the magnitudes of the parameters α and β. It would appear to be exceedingly difficult to derive analytical expressions for these parameters as functions of the constraint length and the rate. Furthermore (56) is only an asymptotic expression that does not purport to be valid for small values of L. To help one get a feeling for the properties of the decoding-lag distribution, especially at small and intermediate lags, a rather limited computer simulation was obtained for the decoding lag experienced at various rates and constraint lengths by a random tree code. The results of this simulation will be reported in section 4.1.

3.2.3 Character of Error Patterns

Another feature of the Viterbi decoding algorithm is that for tree codes with large constraint lengths, decoding errors occur in bursts whose lengths tend to cluster about

some characteristic value. We shall examine that property, and extend similar work by Forney.[11]

Equation 38 is an upper bound over the ensemble of random tree codes on the probability of a decoding error within an unmerged span of length $\nu/(1-\mu)$, where μ is defined by (37). For large ν the bound approaches an equality:

$$P_\mu(\epsilon) \rightarrow \exp -\nu b[e(r, \mu)-0(\nu)], \tag{78}$$

where $e(r, \mu)$ is defined by (39), and $0(\nu)$ goes to zero for large ν. Such a decoding error produces an error sequence of length

$$L_\mu = \frac{\nu}{1-\mu} - \nu = \frac{\nu\mu}{1-\mu}, \tag{79}$$

which is equally likely to be any of the unmerged sequences of that length. Thus, for codes with large constraint lengths, we would expect to observe decoding-error bursts of length L_μ occurring with a probability proportional to $P_\mu(\epsilon)$.

Define μ_o as the μ for which $e(r) = e(r, \mu)$ in (40). Since $e(r)$ is the minimum value of $e(r, \mu)$, we would expect to observe error bursts of characteristic length L_{μ_o} most frequently in the decoding output. Furthermore, the fact that $P_\mu(\epsilon)$ varies exponentially with $e(r, \mu)$ indicates that error bursts of length L_μ are going to occur negligibly often compared with bursts of characteristic length L_{μ_o} whenever μ and μ_o are substantially different.

Define any error burst of length L_μ to be rare if

$$\frac{P_\mu(\epsilon)}{P_{\mu_o}(\epsilon)} \leqslant A, \tag{80}$$

for some arbitrary $A \ll 1$. For large ν, (80) is equivalent to

$$f(\mu) - f(\mu_o) \geqslant \frac{B}{\nu}, \tag{81}$$

where

$$B = \frac{-\ln A}{b}, \tag{82}$$

and

$$f(\mu) = e(r, \mu). \tag{83}$$

Consider first the case $r > R_{comp}$. The quantity $f(\mu)$ can be expanded in a Taylor series about μ_o:

$$f(\mu) = f(\mu_o) + (\mu-\mu_o) f'(\mu_o) + \frac{(\mu-\mu_o)^2}{2} f''(\mu_o) + \ldots . \tag{84}$$

Since $r > R_{comp}$, $f'(\mu_o)$ is identically zero from the definition of μ_o. If third-order and higher order terms in (84) can be neglected, then for large ν (81) becomes

$$(\mu-\mu_o)^2 \geq \frac{2}{f''(\mu_o)} \cdot \frac{B}{\nu}. \tag{85}$$

The range of μ that does not satisfy (85) is proportional to $\nu^{-1/2}$, so that by (79) the range of error burst lengths that are not rare is proportional to $\nu^{1/2}$. Thus for large ν most error bursts have lengths near the characteristic length L_{μ_o}, although the absolute range increases, in the same way that the sum of a large number N of independent random variables clusters closely about N time the mean, although the dispersion is proportional to $N^{1/2}$.

For $r < R_{comp}$, decoding errors tend to occur in short bursts whose distribution of lengths is independent of the constraint length. In this case, $\mu_o = 0$, $f(\mu_o) = E(0)$, and $f(\mu)$ can be given explicitly:

$$f(\mu) = \frac{\left(1 - \frac{\mu r}{R_{comp}}\right) E(0)}{1 - \mu}, \qquad r < R_{comp}. \tag{86}$$

Thus (81) becomes

$$\frac{\mu}{1 - \mu} \geq \frac{C}{\nu}, \tag{87}$$

where

$$C = \frac{B}{\left(1 - \frac{r}{R_{comp}}\right) E(0)}. \tag{88}$$

For sufficiently large ν, $\mu < 1/2$, so that then

$$\mu > \frac{2C}{\nu} \tag{89}$$

is the range of μ for which error bursts are rare. Consequently, the range of μ for which error bursts are not rare is proportional to ν^{-1}, and thus by (79) the range of error burst lengths that are not rare is independent of ν.

For tree codes with large constraint lengths, we would expect for $r > R_{comp}$ that decoding errors would occur in bursts whose lengths cluster around a mean value that is proportional to ν, while for $r < R_{comp}$ decoding errors occur in bursts whose length distribution is essentially independent of ν. It is somewhat harder to predict the burst-length distribution for tree codes with short constraint lengths. A very limited computer simulation of decoding-error statistics (reported in sec. 4.2) tends to suggest the validity of the expected behavior of burst-error patterns.

3.2.4 Decoding Complexity for Systematic Convolutional Codes

For the iterative computational procedure described in section 3.2.1, the number of calculations per branch grows exponentially with both the constraint length ν and the number of symbols per source branch t. This happens because q^t accumulated relative Hamming distances must be computed and compared to determine the minimum-distance path to each of the $q^{\nu t}$ states at each branch. Thus for the algorithm described in section 3.2.1 the decoding complexity is proportional to $q^{(\nu+1)t}$.

It is bad enough that the decoding complexity grows exponentially in ν. Since it usually becomes difficult to build computers that can perform more than 2^{15}-2^{20} calculations per branch in real time, a Viterbi decoder is essentially restricted to using binary codes with constraint lengths not exceeding from 15 to 20. If larger alphabets are used, the allowable constraint length must be reduced accordingly. For example, if q = 4, it appears impractical to build a decoder whose constraint length exceeds 7-10. This restriction on allowable constraint length severely limits the practical usefulness of these codes.

If the decoding algorithm described in section 3.2.1 is used, the fact that the decoding complexity also grows exponentially in t permits the use of only low-rate codes, for which t = 1, in practical systems. If codes in which t > 1 are used, the permissible constraint length must be reduced to the point where it is likely to render the code ineffective for error-correction purposes.

It is possible to modify the algorithm described in section 3.2.1 somewhat in order to achieve a substantial reduction in decoding complexity for high-rate systematic convolutional codes. For high-rate codes, b = t + 1, and only the b^{th} channel system depends on the last νt source symbols plus the current t source symbols. Using (15) and (17), we have

$$x_{ik} = s_{ik}, \qquad k = 1, 2, \ldots, t \tag{90a}$$

$$x_{ib} = \sum_{\ell=0}^{\nu} \sum_{j=1}^{t} g_{jb\ell}^{(i)} s_{(i-\ell)j}. \tag{90b}$$

A circuit for realizing (90) is shown in Fig. 11. It comprises a ν-stage shift register and some multipliers and adders.

Since the contents of the shift register contain all of the information required to characterize the system at each branch, the state of the encoder can be defined by the contents of the shift register. Thus the encoder is defined by q^ν possible states instead of $q^{\nu t}$. At this point, a decoding algorithm exactly like the Viterbi algorithm described in section 3.2.1 can be defined on only these q^ν states. At each branch, each state has q^t possible successor states, so that the total number of distance calculations required at each branch is $q^{\nu+t}$, which is significantly less than $q^{\nu t}$. Like the conventional Viterbi algorithm, the decoder calculates the maximum-likelihood, minimum-distance path to each state, and it decodes all but the last k branches when the q^ν maximum-likelihood

paths are all identical except for the last k branches.

There are some minor differences between this high-rate decoding algorithm and the conventional algorithm, however. In some cases, some of the possible states do

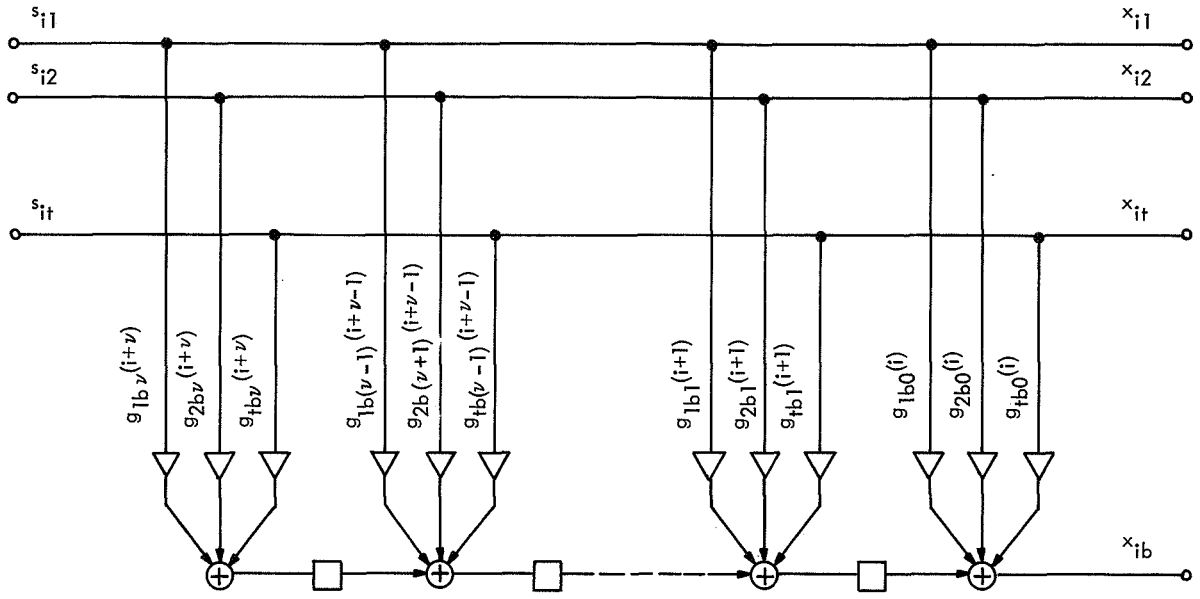

Fig. 11. High-rate systematic convolutional code encoder.

not exist. An example occurs when $g_{jb\nu}(i+\nu) = 0$ for all $j = 1, 2, \ldots, t$, so that all of the possible states except those for which the first shift register stage contains a zero are nonexistent. Second, the sets of predecessor states and successor states may change from branch to branch for a time-variant systematic convolutional code. This property is illustrated by the rate 2/3 systematic binary convolutional encoder shown in Fig. 12. Suppose that at one branch the connections shown by the solid lines are in effect, and suppose that at the next branch the connection indicated by the dotted line is added. Table 1 indicates the encoder output branch and the final state as a function of the initial state and the source branch for the cases in which the dotted line is unconnected and then connected. When the line is unconnected, the predecessor states for the 00 state are the 00 state and the 01 state, while all four states are predecessor states for the 00 state when the line is connected. A third difference between the algorithms is that the path from a state to a given successor state may correspond to two or more distinct source branches. This is again illustrated by Fig. 12 and Table 1 for the unconnected case wherein, for example, the path from the 00 state to the 00 state may correspond to either a 00 or a 01 for the source branch. This situation can lead to rather complicated decoding ambiguities which can be resolved by arbitrarily choosing paths or sequences as in (52). For example, suppose the dotted line in Fig. 12 is unconnected so that the predecessor states for the 00 state are the 00 state and the

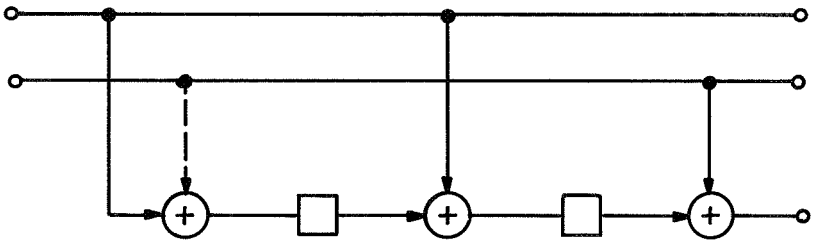

Fig. 12. Rate 2/3 binary systematic convolutional encoder.

Table 1. Outputs and state transitions for encoder of Figure 12.

Initial State	Source Branch	Output Branch	Final State	
			Unconnected	Connected
00	00	000	00	00
	01	011	00	10
	10	100	11	11
	11	111	11	01
01	00	001	00	00
	01	010	00	10
	10	101	11	11
	11	110	11	01
10	00	000	01	01
	01	011	01	11
	10	100	10	10
	11	111	10	00
11	00	001	01	01
	01	010	01	11
	10	101	10	10
	11	110	10	00

01 state, suppose the accumulated relative distance to the 00 state is 2 and to the 01 state is 1, and suppose the branch 111 is received. Let us calculate the minimum-distance path to the 00 state after receiving the branch. From the 00 state the decoder adds a distance of 3 if the 00 source branch is assumed and adds a distance of 1 if the 01 source branch is assumed, so that it chooses the 01 source branch for a total accumulated distance of 3 along the 00-00 path. From the 01 state the decoder adds a distance of 2 if either the 00 or the 01 source branch is assumed, so that by (52) it arbitrarily selects the 00 source branch for a total accumulated distance of 3 along the 01-00 path. But now a decoding ambiguity exists between the 00-00 path and the 01-00 path, since they both lead to total accumulated distances of 3 to the 00 state. By applying (52), again the decoder arbitrarily selects the 01 source branch along the 00-00 path as its final, unique decision, thereby yielding the minimum-distance path to the 00 state.

The Viterbi decoders for low-rate arbitrary tree codes and high-rate systematic convolutional codes have the same expression for decoding complexity per branch, namely $q^{\nu+t}$. For the general case of systematic convolutional codes, let

$$\mu = \inf (t, b-t). \tag{91}$$

Then a Viterbi decoder can be built with a decoding complexity per branch of $q^{\nu\mu+t}$. The decoding complexity is therefore greatest for medium-rate codes. This situation is similar to the decoding complexity observed for block codes, where some high-rate codes such as the single-parity-check codes or the Hamming codes can be easily decoded, some low-rate codes such as the repetition codes or the MacDonald codes can be easily decoded, but where it is usually very difficult to decode large classes of efficient medium-rate codes.

IV. SIMULATION OF THE VITERBI DECODING ALGORITHM

We have presented a reasonably comprehensive analytical treatment of the Viterbi algorithm for decoding tree codes. The algorithm is applicable to any tree code with a finite constraint length, and it is also a maximum-likelihood decoding technique for a large, practical class of channels. Thus the Viterbi algorithm can be applied to any "good" tree code to yield a decoding error probability bounded by (7), the coding theorem bound for tree codes.

Although section 3.2 contains new and interesting results concerning the Viterbi algorithm for decoding tree codes, the analysis is weak in some important practical aspects. We have shown that the asymptotic distribution of the decoding lag is exponentially bounded. This is interesting, of course, and it may give an indication of the form of the tail of the decoding-lag distribution. Nothing in that analysis, however, provided a suggestion of the magnitudes of the relevant parameters, nor did the analysis treat the form of the decoding-lag distribution near the median of the distribution, or show how the decoding-lag distribution varies as a function of ν, R, and C. We have explored the character of decoding-error patterns to be expected in the limiting case of codes with large constraint lengths. As we observed, however, it is usually not practical to apply the Viterbi decoding algorithm to tree codes whose constraint lengths exceed 15-20 branches. It is doubtful to what extent the analysis in section 3.2.3 applies to short, practical tree codes, or what decoding error probability is attainable beyond the bound (7) for these codes. Finally, no suggestion has been made anywhere that the decoding error probability of a particular symbol may be related to the decoding lag of that symbol at the time it is decoded. It would certainly be interesting to determine whether such a relationship exists.

We shall attempt to provide some insight into these questionable areas by analyzing experimental data obtained from a computer simulation of the Viterbi decoding algorithm applied to random tree codes with short constraint lengths. Several topical areas will be explored: (i) we shall examine the gross characteristics of the form of the distribution of the decoding lag as a function of ν, R, and C; (ii) we shall tabulate $\overline{L}$, the average length of a burst of errors, as a function of ν, R, and C, and then relate our observations to the analysis presented in section 3.2.3; (iii) we shall show that the probability of erroneously decoding a particular symbol appears to depend strongly on the decoding lag of the symbol at the time it is decoded; (iv) we shall show that the decoding error probability is approximately 5 times as small as the coding-theorem bound (7); and (v) we shall indicate how our results may be extrapolated to codes whose constraint lengths are longer than those of the codes used in the simulation.

A Digital Equipment Corporation PDP-1 computer was programmed to generate low-rate (normalized rate 1/n) random binary tree codes with arbitrary constraint lengths not exceeding 10 branches, in order to simulate the transmission of these codes over a binary symmetric channel with an arbitrary error probability p, and to decode the

Table 2. Viterbi algorithm simulation data runs.

Run No.	ν	Rate R	p	R/C	Sample Size*
1	5	1/4	0	1/4	10,400
2	8	1/4	0	1/4	10,400
3	10	1/4	0	1/4	5,740
4	5	1/3	0	1/3	9,980
5	8	1/3	0	1/3	9,980
6	10	1/3	0	1/3	4,990
7	5	1/2	0	1/2	20,700
8	8	1/2	0	1/2	10,400
9	10	1/2	0	1/2	5,170
10	5	1/4	.109	1/2	20,700
11	8	1/4	.109	1/2	46,000
12	5	1/9	.230	1/2	20,700
13	5	1/18	.306	1/2	20,700
14	8	1/18	.306	1/2	30,200
15	10	1/18	.306	1/2	15,000
16	5	1/4	.175	3/4	100,000
17	8	1/4	.175	3/4	18,900
18	10	1/4	.175	3/4	10,000
19	5	1/18	.341	3/4	10,300
20	8	1/18	.341	3/4	11,500
21	10	1/18	.341	3/4	10,300
22	5	1/4	.200	9/10	20,900
23	8	1/4	.200	9/10	20,000
24	10	1/4	.200	9/10	15,600
25	5	1/18	.355	9/10	56,800
26	8	1/18	.355	9/10	9,600
27	10	1/18	.355	9/10	5,200
28	5	1/4	.214	1	10,400
29	8	1/4	.214	1	10,300
30	10	1/4	.214	1	5,170
31	5	1/3	.174	1	9,960
32	8	1/3	.174	1	9,940
33	10	1/3	.174	1	4,990
34	5	1/2	.110	1	10,300
35	8	1/2	.110	1	10,300
36	10	1/2	.110	1	5,120
37	5	1/4	.5		10,300
38	8	1/4	.5		10,200
39	10	1/4	.5		5,120
40	5	1/3	.5		10,400
41	8	1/3	.5		9,940
42	10	1/3	.5		5,030
43	5	1/2	.5		10,400
44	8	1/2	.5		10,300
45	10	1/2	.5		5,140

*In source symbols.

resulting channel sequence according to the Viterbi algorithm specified in section 3.2.1. The program computed the probability distribution of the decoding lag, the gross channel and decoding error rates (i.e., fraction of erroneous symbols to total symbols), the distribution of error burst lengths, and the distribution of error-free intervals between bursts. Quite arbitrarily, but reasonably in terms of the computational procedure specified in section 3.2.1, we defined an error burst to be any segment of the decoded source sequence with the following properties: The sequence begins and ends with decoding errors; it contains no error-free subsequences of ν or more consecutive symbols; and it is immediately preceded and followed by error-free intervals of at least ν consecutive symbols. Table 2 lists the data runs that were obtained from this simulation program.

For simplicity, it was assumed in the computer program that the all-zero source sequence was supplied to the encoder, and that the all-zero source sequence produced the all-zero channel sequence. To generate a random tree code, it was further assumed that each of the b symbols assigned to each of the $q^{\nu+1} - 1$ encoder branches not merged with the source sequence at each source branch were chosen independently from a distribution $p(0) = p(1) = 1/2$.

Corresponding to each source branch, the computer generated data simulating the Hamming distance between the received branch and each of the $q^{\nu+1}$ possible encoder branches. The distance between the received branch and the transmitted branch of b "0's" accounted for channel errors. This distance was simulated by choosing b symbols from the distribution $p(1) = p$, $p(0) = 1 - p$, and by totaling all of the "1's" that were selected. To generate a binary symbol from that distribution, the computer used a random-number generator to provide a 13-bit random number, then determined whether this random number exceeded $2^{13}p$, and if not, assigned "1" to the resulting binary symbol.

The distance between the received branch and each of the $q^{\nu+1} - 1$ remaining possible encoder branches was simulated by generating an 18-bit number from the random-number generator, and by counting the number of "1's" in a particular set of b binary digits in that random number. This computational procedure is, of course, perfectly valid for the simulation of a Viterbi algorithm decoder acting on a random tree code applied to a binary symmetric channel, even though it does not correspond to the way in which an actual decoder would work.

The random-number generator produced a periodic pseudo-random sequence of 18-bit numbers. Its period was 109610, a number whose prime factors are 2, 5, 97, and 113. From the description of the manner in which the computer simulated the Hamming distances between the received branch and each of the $q^{\nu+1}$ possible encoder branches, it is evident that the generator was used $q^{\nu+1} + b - 1$ times per source branch. By elementary calculation, one can observe that the random-number generator progressed through many periods for each of the data runs listed in Table 2. Nevertheless, the set of $q^{\nu+1}$ distances simulated for a source branch was different, if not statistically independent, from all other such sets, provided that the random-number generator was

in a distinct state as it began to simulate distances for each source branch. Thus the simulation program would have produced a periodic sequence of $N(\nu, b)$ distinct sets of distances, where $N(\nu, b)$ is equal to 109610 divided by the greatest common divisor of $2^{\nu+1} + b - 1$ and 109610. None of the data runs listed in Table 2 had a sample size exceeding $N(\nu, b)$. While each set of distances generated by the program was different, so that it is reasonable to expect that the data that was simulated provided a good approximation to actual decoder behavior, it is possible that the repeating sequence produced by the random-number generator accounted for some of the anomalies that will be described later.

Once the program had generated the data simulating the Hamming distances between the received branch and each of the $q^{\nu+1}$ possible encoder branches corresponding to a source branch, it then applied the Viterbi decoding algorithm specified in section 3.2.1. Explicitly, the choice (52) was always used to resolve decoding ambiguities, so that the simulation program always resolved decoding ambiguities in favor of the all-zero sequence. Thus the simulation output tended to be slightly optimistic with respect to decoding error probability, and it may have tended to shorten the decoding lag. It would have been more accurate to have used a random choice rather than (52) to resolve decoding ambiguities in the simulation program. We suspect, however, that the inaccuracies introduced by always using (52) had a negligible effect on the gross characteristics of the decoding statistics obtained from the simulation.

For the most part, data runs 1-15 and 28-45 were used to obtain estimates of the distribution of the decoding lag, while data runs 10-36 were used to study the characteristics of decoding-error patterns. All of the data, however, were used to some extent in both parts of the simulation program.

4.1 DISTRIBUTION OF THE DECODING LAG

In order to get an initial idea of the distribution of the decoding lag k, we first simulated a Viterbi decoder for rate 1/4, 1/3, and 1/2 codes with constraint lengths of 5, 8, and 10 branches for three extreme cases: the error-free channel; the channel for which the rate is equal to channel capacity; and the completely random, zero-information channel. These three cases are represented, respectively, by data runs 1-9, 28-36, and 37-45. When the distributions of the decoding lag for the error-free cases were plotted, we found that these distributions depended on both the rate and the constraint length, with short lags being most probable for short, lower rate codes, as might be expected. For the other two cases, however, we found that the decoding lag distributions depended on the constraint length, but that they were essentially independent of the rate.

These observations led us to hypothesize a rather unexpected phenomenon: The distribution of the decoding lag appeared to depend on only two parameters, the constraint length ν, and the ratio R/C of rate to channel capacity.

To test this hypothesis further, we took several longer data runs for random tree

codes with constraint lengths of 5 branches, at rates 1/2, 1/4, 1/9, and 1/18, and for which p was chosen so that the ratio R/C = 1/2. These data were obtained from data runs 7, 10, 12, and 13, respectively. The results were remarkable: as shown in Fig. 13, the distributions of the decoding lag nearly coincided for all four data runs,

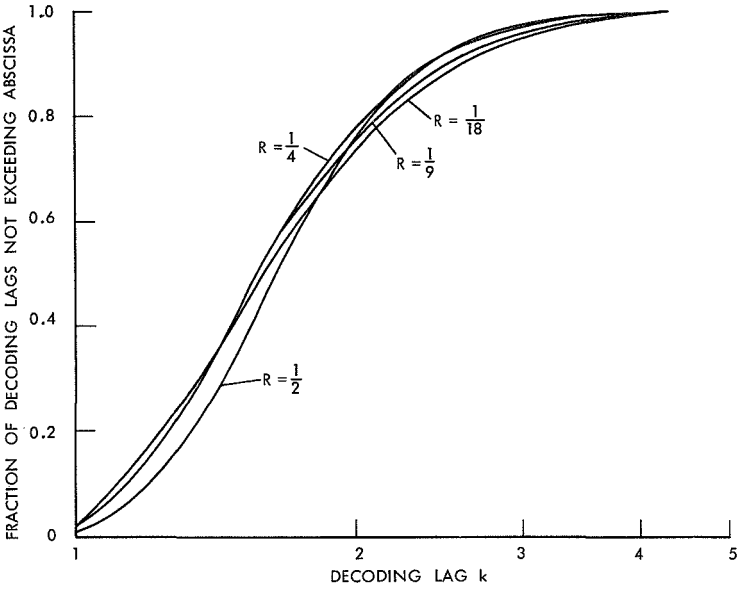

Fig. 13. Perturbations in the distributions of the decoding lags, $\nu = 5$, R/C = .5.

even though the rates were spread over a range of an order of magnitude. Certainly, there were some differences in the four distributions, but these were minor compared with the changes that were observed when either ν or R/C was varied slightly.

The same qualitative effects were observed in comparing the decoding-lag distributions obtained from runs 8, 11, and 14, which were simulations of applying the Viterbi decoding algorithm to random tree codes with constraint lengths of 8 branches, at rates 1/2, 1/4, and 1/18, and for which p was chosen so that R/C = 1/2. Indeed, this phenomenon was noted in the comparison of all subsequent sets of data runs in which the rate could vary, but in which ν and R/C were fixed. Thus we concluded that, as a first-order approximation, the hypothesis was valid, and therefore the distribution of the decoding lag tended to depend primarily on ν and the ratio R/C.

In Figs. 14-16 we have plotted the distribution of the normalized decoding lag k/ν for codes of constraint length 5, 8, and 10, respectively. In each figure the normalized decoding-lag distribution is shown for R/C = 1/2, 1, and infinity. In accordance with the preceding discussion, each distribution that is shown is obtained by combining the data from all data runs for a given ν and R/C. For example, the R/C = 1/2 curve in Fig. 14 represents the combined data from data runs 7, 10, 12, and 13.

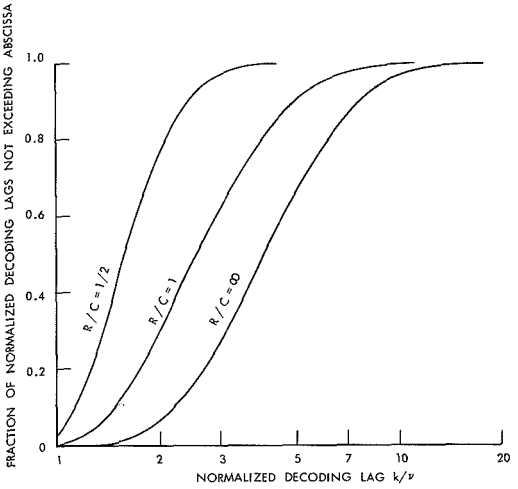

Fig. 14. Distributions of normalized decoding lags, $\nu = 5$.

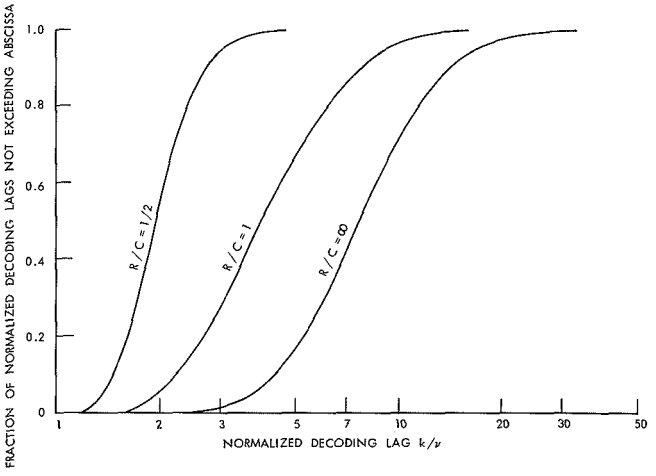

Fig. 15. Distributions of normalized decoding lags, $\nu = 8$.

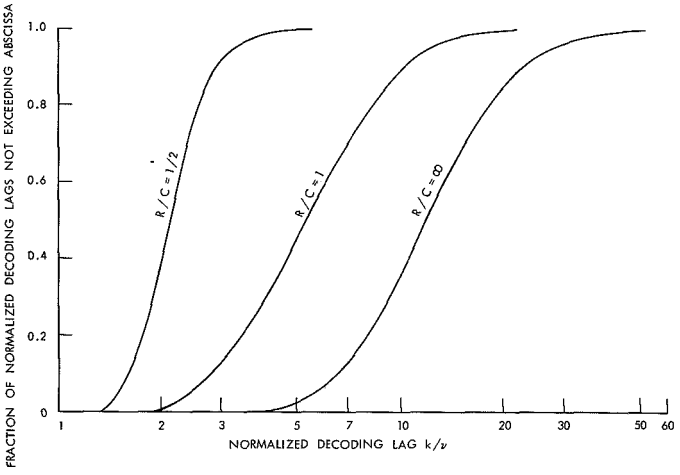

Fig. 16. Distributions of normalized decoding lags, $\nu = 10$.

The distributions shown in Figs. 14-16 have several interesting features. As would be expected, the probability of short decoding lags decreased when either R/C or ν increased. The most interesting observations were quantitative: the logarithm of the median of the distribution of the normalized decoding lag was approximately proportional to $\nu + 1$ and to R/C whenever $R/C \leq 1$, and the logarithm of the median of the $R/C = \infty$ distribution was approximately 1.5 of the logarithm of the median of the $R/C = 1$ distribution for a given ν. Thus

$$\ln M \cong k^* \frac{(\nu+1)R}{C}, \qquad R/C \leq 1. \tag{92}$$

Table 3 lists experimentally determined values of k^* as a function of ν and R/C, including additional data not shown in Figs. 14-16.

Table 3. k^* as a function of ν and R/C.

ν	R/C					
	.25	.33	.50	.75	.90	1.00
5	.159	.161	.154	.160	.158	.157
8	.180	.185	.150	.150	.153	.153
10	.173	.165	.138	.135	.140	.152

Let $F_{R/C,\,\nu}(k/\nu)$ be the distribution function of the normalized decoding lag corresponding to ν and R/C. When we plotted the distributions of the decoding lags for the data runs in which R/C = .25, .33, .50, .75, .90, and 1.00, we found that, as a rough approximation,

$$F_{R/C,\,\nu}(k/\nu) \cong F_{\alpha R/C,\,\nu}[(k/\nu)^{\alpha}] \tag{93}$$

whenever $0 \leq \alpha \leq 1$ and $.1 < F_{R/C,\,\nu}(k/\nu) < .9$. This approximation was especially good near the medians of the distributions, where $F_{R/C,\,\nu}(k/\nu) = .5$.

4.2 ANALYSIS OF DECODING ERRORS

Table 4 lists additional properties not given in Table 2 which further characterize the error patterns of data runs 10-36. In Table 4, the symbols ν, R, p, and R/C are as defined previously, p' is the average decoding error probability, p_B is the coding theorem error probability bound given by (7), $\overline{N}$ and $\overline{L}$ are the average number of errors in a burst and the average burst error length, respectively, and N_B is the total number of error bursts occurring in the data run.

Table 4. Decoding error statistics for data runs 10-36.

Run No.	ν	R	p	R/C	p'	p_B	$\bar{N}$	$\bar{L}$	N_B
10	5	1/4	.109	.50	.0017	.015	2.40	3.14	15
11	8	1/4	.109	.50	.00002	.0012	1	1	1
12	5	1/9	.230	.50	.0045	.024	2.27	2.95	41
13	5	1/18	.306	.50	.0072	.025	1.96	2.39	76
14	8	1/18	.306	.50	.0017	.0027	3.33	5.00	15
15	10	1/18	.306	.50	.0048	.0006	3.42	5.34	21
16	5	1/4	.175	.75	.036	.254	3.94	6.03	944
17	8	1/4	.175	.75	.020	.110	6.85	11.6	54
18	10	1/4	.175	.75	.016	.063	7.80	14.5	20
19	5	1/18	.341	.75	.045	.310	3.30	4.96	135
20	8	1/18	.341	.75	.033	.153	6.77	10.8	56
21	10	1/18	.341	.75	.014	.096	7.44	13.2	19
22	5	1/4	.200	.90	.073	.64	4.78	7.60	320
23	8	1/4	.200	.90	.071	.49	10.0	18.0	143
24	10	1/4	.200	.90	.068	.41	12.7	23.2	84
25	5	1/18	.355	.90	.091	.69	4.32	6.43	1193
26	8	1/18	.355	.90	.093	.55	9.15	16.4	95
27	10	1/18	.355	.90	.078	.47	11.1	20.2	37
28	5	1/4	.214	1.00	.096	1.0	5.33	8.64	187
29	8	1/4	.214	1.00	.123	1.0	13.8	26.0	92
30	10	1/4	.214	1.00	.116	1.0	21.8	33.9	32
31	5	1/3	.174	1.00	.109	1.0	5.95	10.0	183
32	8	1/3	.174	1.00	.096	1.0	11.0	21.1	86
33	10	1/3	.174	1.00	.093	1.0	14.9	26.8	31
34	5	1/2	.110	1.00	.085	1.0	5.13	8.75	164
35	8	1/2	.110	1.00	.109	1.0	13.4	25.9	84
36	10	1/2	.110	1.00	.101	1.0	20.0	36.2	26

Several of the data runs summarized in Table 4 have questionable validity or usefulness. In data run 11, only one source symbol was decoded erroneously from a sample of more than 50,000. Not only is p' much smaller than p_B for this data run, but also the single decoding error gives no indication of the properties of the burst error statistics. In data run 15, the average decoding error probability p' is much larger than the coding theorem bound p_B. It is questionable whether the simulation is an accurate portrayal of the behavior that would be expected operationally in this case. Perhaps these anomalies could be resolved by taking much more data, and perhaps, as we have indicated, they are partly caused by the relatively small period of the

random-number generator that was used in the simulation program. For the most part, however, the data obtained from the simulation agreed reasonably well with the analytical results.

4.2.1 Burst-Error Statistics

Table 5 compares the average burst length $\overline{L}$ obtained from the simulation with the characteristic burst length L_{μ_o} described in section 3.2.3 for data runs 10-36. A comparison is also made of μ_o, obtained by approximating the e(R) curve by

$$e(R) = \begin{cases} R_{comp}, & R \leq R_{comp} \quad (94a) \\ R_{comp} \dfrac{(C-R)}{(C-R_{comp})}, & R_{comp} \leq R < C, \quad (94b) \end{cases}$$

and μ_o^*, obtained by applying $\overline{L}$ to (79):

$$\mu_o^* = \frac{\overline{L}}{\overline{L} + \nu}. \quad (95)$$

By using the approximation (94), it turns out that $\mu_o = R/C$.

To a large extent the simulation data agreed qualitatively with the analytical results given in section 3.2.3. For codes with constraint lengths of 8 or more, the data from runs 14 and 15 indicated that for rates below R_{comp} decoding errors tend to occur in bursts whose lengths are essentially independent of ν, while the remaining data showed that for rates above R_{comp} the average burst length is proportional to ν. These results did not carry over to the $\nu = 5$ codes, thereby suggesting that the analytical results of section 3.2.3 become invalid for codes whose constraint lengths are somewhat less than 8.

There was also an area quantitative disagreement: for codes with constraint lengths of 8 or more, μ_o^* was approximately equal to $.76\,\mu_o$. Thus $\overline{L}$ was much shorter than the characteristic length described previously.

Finally, the data showed that for fixed ν and R/C, $\overline{L}$ decreases with decreasing rate. This suggests a difference between the approximation (94) and the true character of e(R), and it therefore indicates that the approximation $\mu_o = R/C$ is slightly inaccurate. In particular, the simulation data suggest that μ_o increases with decreasing R or increasing channel error probability for a fixed R/C, in agreement with our intuition.

4.2.2 Error Probability as a Function of the Decoding Lag

A limited number of data runs included printouts that enabled the calculation of error probability as a function of the decoding lag. For these data runs,

Table 5. Error burst lengths.

Run No.	$R/C = \mu_o$	μ_o^*	L_{μ_o}	$\overline{L}$
10	.50	.39	§	3.14
11	.50	—	§	1.0
12	.50	.37	§	2.95
13	.50	.32	§	2.39
14	.50	.38	§	5.00
15	.50	.35	§	5.34
16	.75	.53	15	6.03
17	.75	.59	24	11.6
18	.75	.59	30	14.5
19	.75	.50	15	4.96
20	.75	.58	24	10.8
21	.75	.57	30	13.2
22	.90	.60	45	7.60
23	.90	.69	72	18.0
24	.90	.70	90	23.2
25	.90	.56	45	6.43
26	.90	.67	72	16.4
27	.90	.67	90	20.2
28	1.00	.63	∞	8.64
29	1.00	.72	∞	26.0
30	1.00	.77	∞	33.9
31	1.00	.67	∞	10.0
32	1.00	.73	∞	21.1
33	1.00	.73	∞	26.8
34	1.00	.64	∞	8.75
35	1.00	.76	∞	25.9
36	1.00	.78	∞	36.2

§ $R < R_{comp}$ for $R/C \leq .50$.

Table 6 lists the decoding error probabilities of symbols as a function of their decoding lag at the time they were decoded.

The results shown in Table 6 are dramatic. A comparison of the data indicates that the probability of erroneously decoding a symbol, given the decoding lag of the symbol at the time it was decoded, was approximately the same for all five data runs, even though the rates and the ratios R/C varied considerably. The "composite" column in Table 6 lists the error probabilities obtained by combining the data from all five data runs.

Only five data runs were available for this study, since an excessive amount of computer time would have been required to obtain the necessary printouts for all of the data runs. Thus it may be unwise to draw definite conclusions from Table 6. Nevertheless, the table listings suggest that the decoding error probability as a function of the decoding lag is relatively insensitive to R or R/C. If this hypothesis is indeed true, then Table 6 indicates that for a random tree code with $\nu = 10$, the decoding error probability is approximately 10^{-3} when $k = 30$, 3×10^{-3} when $k = 35$, 10^{-2} when $k = 40$, 3×10^{-2} when $k = 50$, 10^{-1} when $k = 60$, and 3×10^{-1} when $k \geq 100$.

Our results suggest that this phenomenon might also be applicable to tree codes

Table 6. Error probability as a function of the decoding lag for codes with $\nu = 10$.

Range of Decoding Lag	Error Probability					
	Run 21	24	30	33	36	Composite
$k < 30$	*	.00036	*	*	*	.00010
$30 \leq k < 35$	.0026	.0051	*	*	*	.0030
$35 \leq k < 40$	.012	.013	.016	.0075	*	.011
$40 \leq k < 45$	.021	.028	.022	.018	.0023	.021
$45 \leq k < 50$	.054	.047	.044	.024	.016	.040
$50 \leq k < 55$	.061	.068	.067	.054	.056	.054
$55 \leq k < 60$	.098	.078	.12	.043	.093	.097
$60 \leq k < 65$	.089	.089	.13	.067	.097	.094
$65 \leq k < 70$	*	.14	.15	.075	.14	.14
$70 \leq k < 75$	*	.19	.20	.074	.14	.17
$75 \leq k < 80$	*	.20	.26	.14	.17	.19
$80 \leq k < 85$	*	.24	.29	.17	.17	.21
$85 \leq k < 90$	*	.22	.34	.19	.11	.21
$90 \leq k < 95$	*	.26	.34	.26	.14	.24
$95 \leq k < 100$	*	.30	.35	.32	.20	.29
$100 \leq k$	*	.29	.37	.30	.31	.31

*No data or insufficient data.

with different constraint lengths. Certainly, the problem is worthy of further experimental study. If, as our data suggest, it is indeed true that the decoding lag and the decoding error probability are highly correlated, then this property may be useful to the designer of a decoding system. The fact that the decoding lag supplies a measure of likelihood information for each decoded symbol might be used, for example, by treating all decoded symbols whose decoding lag exceeds a specified amount as erasures. The property certainly indicates that there is a practical limit to the decoder memory beyond which one gains little in terms of decreasing decoding error probability.

4.2.3 Comparison with the Coding Theorem

By comparing p' with p_B in Table 4, we observe that the decoding error probability is less than the coding theorem bound (7) for every data run except run No. 15. Table 7 lists the ratio of p' to p_B observed as a function of ν and R/C.

Table 7. Ratio of p' to p_B.

R/C	ν		
	5	8	10
.50	.20	*	*
.75	.15	.20	.20
.90	.12	.16	.17
1.00	.10	.11	.11

*Insufficient or unreliable data.

Table 7 indicates that p'/p_B tends to increase with ν and to decrease with R/C, but that it has a value ranging between approximately .15 and .25 for $.50 < R/C < .90$ and $5 \leq \nu \leq 10$.

4.2.4 Projections

In Section V we shall occasionally want to estimate the performance attainable by applying the Viterbi decoding algorithm to tree codes for which $\nu > 10$ and for which $.5 \leq R/C \leq 1.0$. Using the results of sections 4.2.1 and 4.2.3, we conservatively estimate a decoding error probability

$$p' = .25 \exp -\nu be(R), \qquad \nu \leq 20, \tag{96}$$

and we estimate an average burst length

$$\overline{L} = \frac{\nu \mu_o}{1 - \mu_o}, \tag{97}$$

where

$$\mu_o = .76\ R/C, \qquad \nu \geq 8. \tag{98}$$

V. CASCADED TREE CODES

5.1 REVIEW OF BLOCK-CODE CASCADING

In section 1.3 it was indicated that cascading techniques can be used to generate easily decoded codes that have a large constraint length. Cascading is effective whenever, for a given rate and decoding error probability, it yields a substantial reduction in the decoding complexity compared with a one-stage code.

Two effective classes of block-code cascading techniques are product codes and concatenated codes. These techniques, which have somewhat different structures and properties, also have analogs in the formulation of cascaded tree codes. To understand the tree-code cascading techniques, therefore, it is helpful to review the properties of these two classes of block-code cascading techniques.

5.1.1 Product Codes

Elias's[8] invention of product codes was the first application of cascading to the construction of long block codes. The formulation of a two-stage, productlike code can be explained by referring to Fig. 2. The data source supplies symbols from an alphabet of size q to the outer encoder in blocks comprising $k_1 k_2$ symbols each. These symbols can be arranged conceptually in the $k_1 \times k_2$ array shown in Fig. 17a. The outer encoder takes the k_1 symbols from each row and encodes them into blocks of n_1 symbols, where the output symbols also belong to the q-ary alphabet. The outer code can be arbitrary, although in most practical coding systems it is systematic, linear, and time-invariant. The k_2 output blocks comprising n_1 symbols each can be conceptually arranged in the $n_1 \times k_2$ array shown in Fig. 17b. By using an interleaver, the symbols from the $n_1 \times k_2$ array are read into the inner encoder by columns. The inner encoder takes the k_2 symbols from each column of the $n_1 \times k_2$ array and encodes them into blocks of n_2 symbols, where once more the output symbols belong to an alphabet of size q. Again, the inner code format may be arbitrary. The output of the inner encoder can be represented by the $n_1 \times n_2$ array shown in Fig. 17c. The symbols from this array are supplied column by column to the channel.

The decoding process is the reverse of encoding. Each block of n_2 channel output symbols is decoded according to an algorithm appropriate to the inner code to produce an estimate of the k_2 symbols that were supplied to the inner encoder. When a block of n_1 of these k_2-symbol words is decoded, it is arranged conceptually in an $n_1 \times k_2$ array like that shown in Fig. 17b, and an interleaver is used to read the symbols out by rows. Each block of n_1 of these symbols is decoded according to an algorithm appropriate for the outer code, and the output of the outer decoder is the composite decoder's estimate of the symbols originally supplied by the data source.

The extension of this encoding technique to an arbitrary number of stages is obvious. In all cases the composite block length is equal to the product of the block lengths of the constituent codes, while the composite normalized rate is equal to the product of the

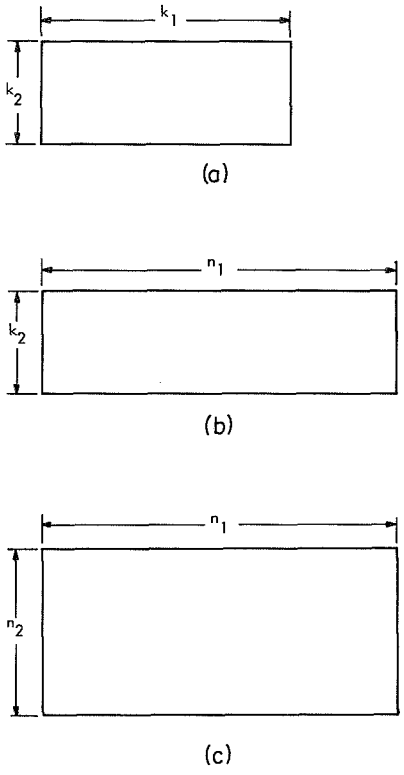

Fig. 17. Representation of product-code symbols.

normalized rates of the constituent codes. [The normalized rate of a code is the fraction of output symbols that represent information. For an (n, k) block code, the normalized rate is k/n.] On the other hand, the composite decoding complexity is approximately equal to the sum of the decoding complexities of the constituent codes, plus the complexity and additional storage requirements introduced by the interleaver.

Elias[8] proposed a product coding system that is remarkably simple. The innermost code is a Hamming single-error-correcting, double-error-detecting code of length 2^m, and each successive code is a similar Hamming code with twice the length of its predecessor. Elias[8] then showed that this coding system could be effectively used on a binary symmetric channel, provided the error probability $\rho < 2^{-(m+1)}$. For such a coding system with s stages, he showed that the decoding-error probability satisfies

$$P(\epsilon) < (2^{m+1}\rho)^{2^s}, \tag{99}$$

while

$$R > 1 - \frac{m+2}{2^{m-1}}. \tag{100}$$

Although (100) is a somewhat loose bound for small values of m, the actual rate of this coding system does fall somewhat short of channel capacity. Furthermore, $P(\epsilon)$ does not decrease exponentially with the composite block length. Finally, the code cannot correct (by this decoding method) some error patterns whose weight is less than half the minimum distance of the code, although it has a substantial diffuse error-correction ability to correct many likely error patterns of weight exceeding half the minimum distance. Despite these shortcomings, this is one of the very few known practical coding systems capable of generating arbitrarily long codes for which the probability of error goes to zero at a reasonable, nonzero rate.

In more recent work that exploits the implementational advantages of cyclic codes, Abramson[27] has designed a cascade decoder for the cyclic product codes introduced

by Burton and Weldon.[28]

All cascaded decoding algorithms for product codes have substantial diffuse error correction, as well as the inability to correct some error patterns whose weight is less than half the minimum distance of the code.

Cascaded algorithms for decoding product codes can realize a substantial reduction in the decoding complexity. If maximum-likelihood decoding is used for decoding each constituent code, for each n_i symbols supplied to the i^{th} decoder, its decoding algorithm must essentially compare $q_i^{ki} = q_i^{n_i r_i}$ alternatives, where q_i is the alphabet size of the i^{th} code, and r_i is the normalized rate of the i^{th} code. For product codes, all constituent codes operate with the same q-ary alphabet. For a product code with s stages of coding, the innermost decoder makes $\frac{1}{n_s} q^{n_s r_s}$ calculations per channel symbol, the next innermost decoder makes $\frac{1}{n_{s-1}} q^{n_{s-1} r_{s-1}}$ calculations per symbol supplied to it, or $\frac{r_s}{n_{s-1}} q^{n_{s-1} r_{s-1}}$ calculations per channel symbol, etc. Thus the total number of calculations per channel symbol for an s-stage product code is

$$\sum_{i=1}^{s} \frac{\prod_{j=i+1}^{s+1} r_s}{n_i} q^{n_i r_i} < \sum_{i=1}^{s} \frac{q^{n_i r_i}}{n_i} \ll \frac{q^{\prod_{i=1}^{s} n_i r_i}}{\prod_{i=1}^{s} n_i}, \tag{101}$$

where $r_{s+1} \triangleq 1$ in (101), and the right-hand expression in (101) is the number of calculations per channel symbol required for a single-stage maximum-likelihood decoder with the same composite length and rate as the product code.

5.1.2 Concatenated Codes

Forney's[9,10] concatenated codes are a second cascading technique for generating long block codes. Concatenated codes and product codes are quite different in their structures and properties, except that the encoding and decoding processes for both classes of codes involve a series of successive, cascaded operations.

In terms of Fig. 2, the construction of concatenated codes can perhaps be most easily understood by considering first the inner code, and then the successively cascaded stages of outer codes. The inner code has a block length n_1 and a normalized rate k_1/n_1, and operates on symbols from a channel alphabet of size q_c. This code can be arbitrary, but in most practical systems it would usually have some mathematical structure such as time invariance, linearity, or being systematic.

Each inner code word represents one of $q_c^{k_1}$ possibilities, which can be interpreted as a symbol in an alphabet of size $q_2 = q_c^{k_1}$. Using this interpretation, the outer code

has block length n_2 and a normalized rate k_2/n_2, but operates on the derived q_2-ary alphabet.

The cascading process can be extended in an obvious manner to an indefinite number of stages. For a concatenated code with s stages of cascading, the source supplies symbols that are interpreted by the outer encoder as blocks of k_s symbols each from a q_s-ary alphabet, where $q_i = q_c^{\prod_{j=1}^{i} k_{j-1}}$, $k_o = 1$, $i = 1, 2, \ldots, s$. The outermost encoder encodes these symbols into blocks of n_s q_s-ary symbols. Each of these q_s-ary symbols is then encoded by the next outermost encoder into blocks of n_{s-1} q_{s-1}-ary symbols, and the process is continued through the innermost encoder that encodes each q_2-ary symbol into blocks of n_1 q_c-ary symbols. Observe that, in contrast to product coding, no interleaver is required between successive stages of coding for concatenated codes.

Again the decoding process is the reverse of encoding. The innermost decoder forms estimates of the q_2-ary symbols, the next innermost decoder uses the q_2-ary symbols estimated by the innermost decoder to form estimates of the q_3-ary symbols, and so on until the outermost decoder finally estimates the k_s q_s-ary symbols originally supplied by the source.

As with product codes, the block length of a composite concatenated code, expressed in channel symbols, is equal to the product of the block lengths of the constituent codes in their respective alphabets, while the normalized rate of the composite code is equal to the product of the normalized rates of the constituent codes. Similarly, also, the composite decoding complexity is approximately equal to the sum of the decoding complexities of the constituent codes. This last statement does not by itself imply that concatenation can substantially reduce the decoding complexity in the way that (101) implies that product-code cascading reduces decoding complexity, since the alphabet sizes differ substantially for concatenated codes. It takes further work to show that concatenation can offer a reduction in decoding complexity compared with the complexity required to decode a single-stage code of the same length as the composite code.

Forney[9,10] has demonstrated a concatenated coding system for which the probability of error decreases nearly exponentially with the composite length at all rates below capacity, while the decoding complexity is proportional to the cube of the composite length, where the constant of proportionality depends on the composite rate. The coding system has a suitably chosen binary BCH code as the inner code, with outer Reed-Solomon codes whose lengths and alphabet sizes are determined by the inner code. For this system the decoding-error probability decreases essentially exponentially with the complexity in the limit of high complexities. The efficiency of these codes – that is, the ratio of the effective concatenated error exponent to E(R) – is quite low, however, especially at rates approaching capacity.

Despite their shortcomings, cascaded block-coding techniques are an effective, practical, and general approach to the coding problem.

5.2 FORMULATION OF CASCADED TREE CODES

We have outlined the structure and properties of product codes and concatenated codes. Both of these coding techniques involve only block codes, and both techniques are an effective approach to the coding problem.

In Section II we showed that in some ways tree codes are superior to block codes. It is natural to wonder whether cascading techniques can be applied to coding systems involving tree codes and, if so, whether the use of tree codes offers any advantage in performance compared with block-code cascading.

Pinsker[29] and Stiglitz[30] have studied two-stage cascaded coding systems in which the inner code is a block code and the outer code is a tree code. Their objective was to discover under what conditions such a coding system could be used so that sequential decoding could be applied to the outer code while the composite rate was higher than R_{comp} for the raw channel. Some of their results will be reported here. It was not one of their primary objectives, however, to study the efficiency and error-correcting capabilities of these codes, nor to study alternative tree-code decoding techniques. In similar work Falconer[31] proposed a two-stage coding scheme in which the inner code comprised N tree codes operating in parallel. K of these codes were independent; the symbols for the remaining N-K codes were specified by an (N,K) block code, which could be regarded as an outer code. Each tree code was decoded independently by using sequential decoding; however, when K tree codes had been decoded, the block code format was applied to decode the remaining N-K tree codes. The primary objective of this technique was to be able to use sequential decoding at composite channel rates exceeding R_{comp}.

In the rest of Section V we shall consider the properties of cascaded coding systems in which both the inner code and the outer code in Fig. 2 are tree codes. Several alternative formulations of cascaded tree codes will be given. Some of these techniques are analogous to the product codes and the concatenated codes, and some are distinctly different. We shall compare the efficiency of tree-code cascading to the efficiency attainable with block-code cascading. It is indicated that in this respect tree-code cascading is greatly superior to block-code cascading, especially at rates approaching channel capacity. Finally, we shall consider the practicality of tree-code cascading, using present available techniques for decoding the constituent codes. Unfortunately cascading, at present, offers at best a marginal improvement. Conditions will be presented for which a cascaded coding system can employ sequential decoding on the outer code of a two-stage system, while operating at a composite rate exceeding R_{comp} for the channel.

5.2.1 Productlike Codes

Suppose that both codes in the two-stage cascading process illustrated in Fig. 2 are tree codes. Let the outer encoder be characterized by parameters ν_1, t_1, b_1, and q, while

the inner encoder is characterized by parameters ν_2, t_2, b_2, and q, where the ν's are the constraint lengths, the t's and the b's are the input and output branch lengths, respectively, and q is the symbol alphabet size for each code. With a suitable interleaver inserted between the two stages of encoding, the composite encoder configuration of an outer encoder, interleaver, and inner encoder generates a two-stage cascaded tree code that is analogous to product codes.

First, consider the amount of symbol separation that must be provided by the interleaver. Since the constraint length of a tree code means that each output symbol "depends on" the last $\nu+1$ input branches supplied to the encoder, then for a two-stage productlike tree code each channel symbol should "depend on" $(\nu_1+1)(\nu_2+1)$ data source branches. This dependence is ensured if a $[(\nu_2+1)t_2, (\nu_1+1)b_1]$ interleaver is inserted between the outer encoder and the inner encoder. [An interleaver for which the symbols in every contiguous set of n_2 symbols in the output sequence were mutually separated by at least n_1 symbols in the input sequence is called an (n_2, n_1) interleaver. In Section VI we discuss the efficient realization of (n_2, n_1) interleavers.] The function of the interleaver is to reorder the output sequence from the outer encoder in such a manner that all of the $(\nu_2+1)t_2$ symbols appearing within the constraint span of the inner encoder "depend on" distinct sets of $(\nu_1+1)t_1$ data source symbols. Thus the constraint length in source branches of the composite code is $(\nu_1+1)(\nu_2+1)t_2 - 1$. Viewed from the data source, every data symbol affects $(\nu_1+1)b_1$ output symbols from the outer encoder, each of which influences a distinct set of $(\nu_2+1)b_2$ channel symbols. Thus the channel symbol constraint length is essentially $(\nu_1+1)(\nu_2+1)b_1b_2$.

The composite encoding and decoding procedures are what one would naturally expect for cascaded codes. The data source symbols are supplied to the outer encoder, which encodes them in its format at a normalized rate t_1/b_1. The output symbols from the outer encoder are interleaved and then supplied to the inner encoder, which encodes them in its format at a normalized rate t_2/b_2. Decoding is just the reverse of encoding. The channel output symbols are supplied to the inner decoder, and its output sequence is unscrambled and supplied to the outer decoder. The output of the outer decoder is the composite decoder's estimate of the data sequence.

The amount of symbol separation provided by a $[(\nu_2+1)t_2, (\nu_1+1)b_1]$ interleaver is only a minimum amount required to obtain a composite constraint length of $(\nu_1+1)(\nu_2+1)t_2 - 1$. Any (n_2, n_1) interleaver with $n_2 \geq (\nu_2+1)t_2$ and $n_1 \geq (\nu_1+1)b_1$ will do. Some decoding algorithms are effective only if the input errors appear to occur randomly rather than in bursts. To use these algorithms effectively in the outer decoder, therefore, it is necessary for the interleaver to provide much greater symbol separation; that is, to make $n_2 \gg (\nu_2+1)t_2$ if the inner decoder has a substantial probability of making errors in bursts of more than $(\nu_2+1)t_2$ symbols whenever it does make decoding errors. The only effect of this modification on decoding complexity is to increase the required symbol storage.

It is almost trivial to extend this coding procedure to the construction of multistage

productlike cascaded tree codes. Under the assumption that sufficient interleaving is inserted between successive stages of encoding, the composite constraint length for an s-stage productlike code is

$$\nu = \left[\frac{1}{t_1} \prod_{i=1}^{s} (\nu_i+1)t_i\right] - 1, \tag{102}$$

while the composite constraint length in channel symbols is

$$\nu_c = \prod_{i=1}^{s} (\nu_i+1)b_i. \tag{103}$$

The normalized rate is just the product of the normalized constituent rates:

$$r = \prod_{i=1}^{s} t_i/b_i \tag{104}$$

Since the alphabet sizes are the same for all of the constituent codes of a productlike code, we would expect that the cascade decoding of these codes would yield a substantial decrease in the decoding complexity compared with the complexity required to decode a single-stage code of the composite constraint length such as that indicated in (101).

Because of the continuous nature of tree codes, there is no need to provide branch synchronization between successive stages of coding. In this regard, the parameters ν_1, ν_2, t_1, t_2, b_1, and b_2 may be chosen independently of each other. Furthermore, there are no restrictions on the parameters n_1 and n_2 of the (n_2, n_1) interleaver inserted between the outer encoder and the inner encoder, except that they be large enough to provide sufficient symbol separation to achieve a composite constraint length equal to the product of the constituent constraint lengths. This contrasts with the analogous block-code formulation, where branch synchronization and much more restrictive or explicit interleaving functions are required to achieve the multidimensional array structure illustrated in Fig. 17.

5.2.2 Concatenationlike Codes

Suppose now that the outer and inner codes in the two-stage cascading process are tree codes with parameters ν_1, t_1, b_1, and q^m, and ν_2, t_2, b_2, and q, respectively. It is evident that one symbol in the outer code alphabet can be transformed into m symbols in the inner-code alphabet, and vice versa. Using that fact, the equipment configuration shown in Fig. 18 can be used to generate a two-stage cascaded tree code that is analogous to a concatenated code.

It is convenient to assume that the data source supplies q^m-ary symbols to the outer encoder. These symbols are encoded by the outer encoder in its format

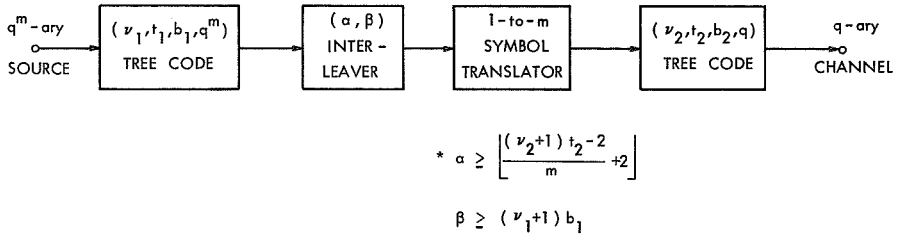

Fig. 18. Encoding for concatenationlike tree codes.

at a normalized rate t_1/b_1. The resulting q^m-ary symbols are fed to an $[\alpha, (\nu_1+1)b_1]$ interleaver that operates in the q^m-ary alphabet, where

$$\alpha = \left\lfloor \frac{(\nu_2+1)t_2 - 2}{m} + 2 \right\rfloor, \tag{105}$$

and "$\lfloor x \rfloor$" means "the greatest integer contained in x." The reordered sequence of q^m-ary symbols are supplied to a q^m-ary-to-q-ary translator: each q^m-ary symbol is transformed into a sequence of m q-ary symbols. The resulting sequence of q-ary symbols is fed to the inner encoder, which encodes these symbols in its format at a normalized rate t_2/b_2. The function of the interleaver is to reorder the output sequence from the outer encoder so that any symbols within the constraint length of the inner encoder that come from the transformation of distinct q^m-ary symbols depend on distinct sets of $(\nu_1+1)t_1$ data source symbols. Viewed from the data source, each data symbol affects $(\nu_1+1)b_1$ output symbols from the outer encoder, and each of these affect a distinct set of at least $(\nu_2+1)b_2$ channel symbols. Thus the channel symbol constraint length is at least $(\nu_1+1)(\nu_2+1)b_1b_2$.

As one would expect, decoding is just the reverse of encoding. The channel output symbols are supplied to the inner decoder, and the resulting sequence of decoded q-ary symbols is converted into a sequence of q^m-ary symbols which is unscrambled and then supplied to the outer decoder. The sequence of output symbols from the outer decoder is the composite decoder's estimate of the data sequence.

The $[\alpha, (\nu_1+1)b_1]$ interleaver provides only the minimum amount of symbol separation necessary for a composite channel symbol constraint length equal to the product of the constituent output symbol constraint lengths. Whenever the inner-code decoding-error distribution coupled with the outer code decoding method requires further symbol separation, any (n_2, n_1) interleaver with n_2 sufficiently greater than some $\alpha_1 \geq \alpha$ and n_1 sufficiently greater than some $\beta \geq (\nu_1+1)b_1$ will suffice.

The extension of this coding procedure to the construction of a multistage concatenationlike code is straightforward. The alphabet size for the i^{th} stage of coding from the channel is $q^{m(i)}$, where for an s-stage concatenationlike code,

$$m(1) = 1 \tag{106a}$$

$$m(i) = \prod_{j=2}^{i} m_{j-1} \qquad i = 2, 3, \ldots, s. \tag{106b}$$

The data source supplies $q^{m(s)}$-ary symbols to the outermost encoder. These symbols are encoded and then interleaved, and are then translated into blocks of m_{s-1} $q^{m(s-1)}$-ary symbols. The sequence of $q^{m(s-1)}$-ary symbols is fed to the second outermost encoder, and the process continues in an obvious manner until the innermost decoder supplies a sequence of q-ary symbols to the channel. Decoding is again just the reverse of encoding. The composite normalized rate is again given by (104), while (103) is a lower bound on the composite constraint length in channel symbols, provided there is sufficient symbol separation between successive stages.

As with the productlike codes, it is unnecessary to provide branch synchronization between successive stages of coding, except that framing synchronization must be provided to the $q^{m(i-1)}$-ary-to-$q^{m(i)}$-ary symbol translators. Also the $\{\nu_i\}$, $\{t_i\}$, and $\{b_i\}$ may be chosen independently of each other, with the obvious restriction that $b_i \geq t_i \geq 1$ for all i. Additionally, the parameters $\left\{n_1^{(i)}\right\}$ and $\left\{n_2^{(i)}\right\}$ for the interleavers between the i^{th} and $(i+1)^{th}$ stages may be chosen arbitrarily, provided they are sufficiently large to furnish the required symbol separation. Finally, in contrast to Forney's[10] concatenated codes, the $\{m_i\}$ that define the alphabet sizes of each stage may be chosen arbitrarily from the positive integers without regard to any of the other sets of parameters previously mentioned. [The use of sequential decoding for the outermost stage or stages may restrict the $\{m_i\}$ in terms of the $\{\nu_i\}$. This topic will be treated in section 5.4.2.] This flexibility in the choice of parameters differs considerably from the analogous situation with block-code concatenation.

There is one final difference between block-code concatenation systems and tree-code concatenation systems. The encoding system for a tree code, shown in Fig. 18, uses an interleaver, while the corresponding block-code concatenation system does not. If there were no interleaver between successive stages, and if m were not proportional to $\nu_2 b_2$, then the approximate composite constraint length in channel symbols for a two-stage code would be

$$\nu_c \cong \left\{ \left\lfloor \frac{(\nu_1+1)b_1 m}{t_2} \right\rfloor + \nu_2 \right\} b_2, \tag{107}$$

so that the composite constraint length would grow more like the sum rather than the product of the constraint lengths of the constituent codes. The interleaver ensures productlike growth, whenever m is not proportional to $\nu_2 b_2$. In the block-code concatenation formulation, m is proportional to the block length of the inner code, so that the productlike growth in block length is automatically provided without the interleaver.

5.2.3 Burst-Error-Correcting Outer Codes

There is a third, reasonably obvious, formulation for a tree-code cascading system that has no clearly defined block-code cascading analog. This formulation makes use of the fact that decoding errors for tree codes tend to occur in bursts, as is indicated in section 3.2.3 and is further supported by the simulation results given in section 4.2.1.

Consider once more the two-stage coding process illustrated in Fig. 2. Let the inner code be a tree code characterized by parameters ν_2, t_2, b_2, and q. Furthermore, assume that the statistical properties of the inner code decoding errors, such as burst-length distributions, error-free interval distributions, and perhaps burst-error densities, can be described and calculated in some suitable manner. Then it may be appropriate to use a burst-error-correcting tree code as the outer code. This code would be chosen to match as efficiently as possible the statistics of the decoding errors of the inner code. Some examples of families of burst-error-correcting convolutional codes are given by Gallager.[3]

Whenever this cascaded system is appropriate, it can yield improvement with relatively little decoding complexity. For example, no interleavers are required between successive stages, and the realization of the burst-error-correcting outer decoders is fairly simple. Like the other classes of cascaded tree codes, the composite normalized rate is equal to the product of the normalized rates of the constituent codes. It would be possible to define a composite constraint length for this class of codes which would be sumlike, as in (107); however, this concept makes little sense because the outer code is a specified, burst-error-correcting code that is matched to the error statistics of the inner-code decoder. Instead it is more meaningful to evaluate this coding system in terms of the error probability attainable at a given composite rate and composite decoding complexity.

Several techniques are possible for extending this basic coding system to a multistage cascaded tree code. Given a multistage cascaded tree code in which the innermost code is a random-error-correcting code that produces decoding errors in well-defined bursts, and is immediately followed by a burst-error-correcting tree code that is matched to the inner-code error statistics, then it is possible to envision at least three alternative possibilities for the next stage of coding: In one case, it might be possible to effectively use a second burst-error-correcting tree code that is matched to the decoding errors of the innermost burst-error-correcting tree code. On the other hand, it might be preferable to use an interleaver and to form a productlike code with a third stage, or to make an m-to-1 symbol translation and then to use an interleaver to form a concatenationlike code with the third stage. Furthermore, it is evident that these three choices are present at each further stage of coding. It is rather difficult to compare any of these techniques without reference to a specific application, since their relative performances might vary widely.

5.3 EFFICIENCY OF TWO-STAGE CASCADING

5.3.1 Coding Theorem Efficiency

Suppose that the channel in the two-stage cascaded coding system shown in Fig. 2 is a DMC with block-code exponent E(R) and tree-code exponent e(R), where the relationship between E(R) and e(R) is given by (39) and (40) and is illustrated in Fig. 6. Let α be the composite constraint length of the cascaded code, where α is the number of channel symbols whose selection is influenced by any given data source symbol. For example, if both codes are block codes with inner-code block length N_2 and outer-code block length N_1, then $\alpha = N_1 N_2$ for a product code. We shall define α for the case in which the inner code is a tree code in the discussion immediately preceding Theorem 4 below. For any combination of block codes and tree codes in the two-stage cascaded coding system, it is possible to find a coding system for which

$$P(\epsilon) < \exp -\alpha\left[E^{\beta_1\beta_2}(R) - o(\alpha) \right], \tag{108}$$

where $o(\alpha) \to 0$ for large α, $\beta_i = T$ if the i^{th} code is a tree code, $\beta_i = B$ if the i^{th} code is a block code, and

$$E^{\beta_1\beta_2}(R) > 0, \qquad 0 \leq R < C. \tag{109}$$

The quantity $E^{\beta_1\beta_2}(R)$ is called the cascading exponent for the channel corresponding to the assignment of β_1 and β_2. Forney[9] has demonstrated the existence of $E^{BB}(R)$; it follows immediately that $E^{TB}(R)$ exists, and can be found from $E^{BB}(R)$ by using (39) and (40) or the graphical technique shown in Fig. 6. In proving Theorem 4 below, we use Forney's[9] arguments to demonstrate the existence of $E^{BT}(R)$, and thus also, as a consequence, the existence of $E^{TT}(R)$.

The ratio

$$\eta^{\beta_1\beta_2}(R) = \begin{cases} E^{B\beta_2}(R)/E(R), & \beta_1 = B \\ E^{T\beta_2}(R)/e(R), & \beta_1 = T \end{cases} \tag{110}$$

is called the efficiency of cascading. Its reciprocal indicates approximately how much longer the composite constraint length of a cascaded code must be than the constraint length of a single-stage code for the cascaded code to attain the same error probability as the single-stage code.

Theorem 4 below asserts that if a tree code is used for the inner code of a two-stage cascaded coding system, it is possible to attain an efficiency that is independent of the

rate for rates below channel capacity. The theorem applies only to concatenationlike codes in which blocks of

$$m = \psi \nu_2 t_2 \tag{111}$$

contiguous q-ary symbols from the output of the inner decoder are used to form symbols in a q_1-ary alphabet used by the outer code, where

$$q_1 = q^m, \tag{112}$$

and the inner code is characterized by the parameters ν_2, t_2, b_2, and q. The parameter ψ defined by (111) indicates the degree of concatenation of the cascaded code. Each set of t_2 q-ary symbols supplied to the input of the inner encoder influences the selection of $(\nu_2+1)b_2$ channel symbols, so that each q_1-ary symbol influences the selection of $(\psi+1)\nu_2 b_2$ channel symbols. If an interleaver is used between the stages of cascading, as in Fig. 18, so that each q_1-ary symbol influences a distinct set of $(\psi+1)\nu_2 b_2$ channel symbols, then it is evident that $a = N_1(\psi+1)\ \nu_2 b_2$ if the outer code is a block code, and $a = (\nu_1+1)b_1\ (\psi+1)\nu_2 b_2$ if the outer code is a tree code.

Theorem 4

In a two-stage concatenationlike coding system in which the inner code is a tree code characterized by the parameters ν_2, t_2, b_2, and q, and the alphabet size of the outer code is given by (111) and (112), for arbitrary β_1,

$$\eta^{\beta_1 T}(R) = \frac{1}{2}, \qquad 0 \leq R < C, \tag{113}$$

provided $\psi = 1$ in (111).

Proof of Theorem 4: It suffices to specify a coding system for which

$$E^{BT}(R) = \frac{1}{2}\, E(R), \tag{114}$$

since $E^{TT}(R)$ can be constructed from $E^{BT}(R)$ by using (39) and (40), or Fig. 6, and $E^{TT}(R) \to \frac{1}{2}\, e(R)$ if $E^{BT}(R) \to \frac{1}{2}\, E(R)$.

The rate of the inner code is

$$R_2 = \frac{t_2}{b_2} \ln q, \tag{115}$$

and its decoding error probability per symbol is bounded by

$$P(\epsilon) < \exp -\nu_2 b_2 [e(R_2) - o(\nu_2 b_2)]. \tag{116}$$

Using a union bound, the probability that a q_1-ary symbol supplied to the outer decoder is erroneous is bounded:

$$p < \psi \nu_2 t_2 \exp -\nu_2 b_2 [e(R_2) - o(\nu_2 b_2)]$$

$$= \exp -\nu_2 b_2 \left[e(R_2) - \frac{1}{\nu_2 b_2} \ln \psi \nu_2 t_2 - o(\nu_2 b_2) \right]$$

$$= \exp -\nu_2 b_2 [e(R_2) - o'(\nu_2 b_2)], \tag{117}$$

where $o'(\nu_2 b_2) \to 0$ for large $\nu_2 b_2$. If the inner code is decoded according to the Viterbi decoding algorithm, the discussion in section 3.2.3 indicates that decoding errors occur in bursts whose lengths cluster around some characteristic length that is proportional to ν_2 for $R_2 > R_{comp}$. Thus if ψ is carefully chosen, (48) gives a close approximation to the marginal q_1-ary symbol transition probabilities:

$$p_{ij} = 1 - p, \qquad i = j \tag{118a}$$

$$= \frac{p}{q_1 - 1}, \qquad i \neq j. \tag{118b}$$

Forney[10] has shown that the q_1-ary discrete memoryless channel whose transition probabilities are given by (118) is the worst of all q_1-ary DMC's, in the sense that its error exponent is the lowest for a given p. Thus our results, derived for the channel defined by (118), will hold for any q_1-ary DMC.

Now let the outer code be a block code of length N and normalized rate μ whose symbols are taken from the q_1-ary alphabet. By approximately interleaving the q_1-ary symbols from the output of the inner decoder, the symbols within a block can be made to appear statistically independent, so that the superchannel seen by the outer decoder looks like a DMC, with transition probabilities given by (118). Then (2) can be used to bound the probability that an outer code block is erroneously decoded:

$$P(\epsilon) < \exp -N E_1(R'), \tag{119}$$

where, in this case,

$$R' = \mu \ln q_1 = \nu_2 b_2 \psi \mu R_2, \tag{120}$$

and, from (3),

$$E_1(R') = \sup_{0<\rho<1} [E_o(\rho) - \rho R']. \tag{121}$$

Using (4) and (118), we obtain

$$E_o(\rho) = -\ln q_1^{-\rho}[1-\rho)^{1/(1+\rho)} + (q_1-1)^{\rho/(1+\rho)} p^{1/(1+\rho)}]^{1+\rho}. \tag{122}$$

When ν_2 is sufficiently large, one or the other of the two terms in brackets in (122) dominates, so that with an error of at most ln 2,

$$E_o(\rho) = \begin{cases} \nu_2 b_2 \rho\psi R_2, & \rho\psi R_2 < e^*(R_2) \\ \nu_2 b_2 e^*(R_2), & \rho\psi R_2 > e^*(R_2) \end{cases}$$

$$= \nu_2 b_2 \min \{\rho\psi R_2, e^*(R_2)\}, \tag{123}$$

where $e^*(R) = e(R) - o'(\nu_2 b_2)$ from (117). Equation 122 is maximized by setting $\rho = \min \{1, e^*(R_2)/\psi R_2\}$. Thus

$$E_1(R') = \nu_2 b_2 E^*(\mu, R_2), \tag{124}$$

where we define

$$E^*(\mu, R_2) = (1-\mu) \min \{\psi R_2, e^*(R_2)\}. \tag{125}$$

Equations 119, 120, 124, and 125 can be combined to form a bound for $P(\epsilon)$ that can be expressed in terms of the composite rate R' seen by the q_1-ary symbols. We shall now derive a bound for $P(\epsilon)$ to be expressed in terms of the composite rate R that applies to the q-ary channel symbols. Finally, we shall relate this last bound to the composite constraint length of the code in channel symbols, which will be used to evaluate the efficiency $\eta^{BT}(R)$, and thereby complete the proof of Theorem 4.

Continuing along this line, therefore, we define

$$E^{BT*}(R) = \sup E^*(\mu, R_2), \tag{126}$$

where

$$R = \mu R_2 \tag{127}$$

is the composite rate that applies to the q-ary channel symbols, and $E^*(\mu, R_2)$ is given by (125). Figure 19 illustrates the construction of $E^{BT*}(R)$ from $e^*(R)$ whenever $\psi \geq 1$. Curves are drawn showing $E^*(\mu, R_2)$ as a function of $R = \mu R_2$ for 5 fixed values of R_2. The $E^{BT*}(R)$ curve is the upper envelope of all possible $E^*(\mu, R_2)$ curves. Since the $E^*(\mu, R_2)$ curves lie below the $E^*(\mu, R_{comp})$ curve for $R_2 < R_{comp}$, it is clear that $R_2 \geq R_{comp}$ in the maximization of (126) whenever $\psi \geq 1$. Using (125) and (126), we obtain

$$E^{BT*}(R) = \sup_{\substack{0 \leq \mu < 1 \\ R_2 \geq R_{comp}}} (1-\mu)\, e^*(R_2). \tag{128}$$

But (128) is just like (40) in which the unknown tree-code exponent e(R) is expressed in terms of E(R), which is known in deriving (40). By comparing (128) with (40), it is clear that $E^{BT*}(R) = E(R) - o''(\nu_2 b_2)$ whenever $\psi \geq 1$, so that

$$P(\epsilon) < \exp -N\nu_2 b_2 [E(R) - o''(\nu_2 b_2)]. \tag{129}$$

A similar procedure can be used to construct $E^{BT*}(R)$ when $\psi < 1$. It is easier graphically to understand the construction of $E^{TT*}(R)$ instead, which is illustrated in Fig. 20. Defining $R_E(\psi)$ as the value of R_2 for which $e(R_2) = \psi R_2$, we find that for $R \geq R_E(\psi)$, $E^{TT*}(R) = e(R)$, and for $R \leq R_E(\psi)$, $E^{TT*}(R) = e[R_E(\psi)]$.

We now relate $E^{BT*}(R)$ to the concatenation exponent $E^{BT}(R)$. We want a bound of the form

$$P(\epsilon) < \exp -N(\psi+1)\, \nu_2 b_2 [E^{BT}(R) - o(\nu_2 b_2)]. \tag{130}$$

Comparing (130) with (129), we see that

$$E^{BT}(R) = \frac{E(R)}{\psi + 1}, \qquad \psi \geq 1, \quad 0 \leq R < C. \tag{131}$$

By choosing $\psi = 1$, Theorem 4 is proved.

Theorem 4 shows that it is possible to find a cascading system in which the inner code is a tree code for which the efficiency is constant at all rates below capacity. This contrasts sharply with Forney's[9] results, which indicate that for a two-stage cascading system in which the inner code is a block code, the efficiency decreases monotonically with rate, and becomes arbitrarily small at rates approaching capacity. As we shall see, however, the cascaded tree codes that are most efficient in terms of the coding theorem are sometimes quite inefficient in their ability to reduce decoding complexity.

The results of Theorem 4 can be extended to an n-stage cascaded coding system in which all of the codes are tree codes. Let the symbols for the $(i+1)^{th}$ innermost code be obtained by grouping $\psi_i \nu_i t_i$ symbols from the i^{th} innermost code, $i = 1, 2, \ldots, n-1$. Let $\psi_i = \frac{1}{n-1}$, $i = 1, 2, \ldots, n-1$. Then it turns out that the composite decoding error probability of the n-stage code analogous to (129) is given by

$$P(\epsilon) < \exp -\left(\prod_{i=1}^{n-1} \nu_i b_i\right) \left[E^{TT*}(R) \Big|_{\psi = \frac{1}{n-1}} - o(\pi) \right], \tag{132}$$

where the construction of $E^{TT*}(R)$ from e(R) is shown in Fig. 20 for $\psi < 1$. The

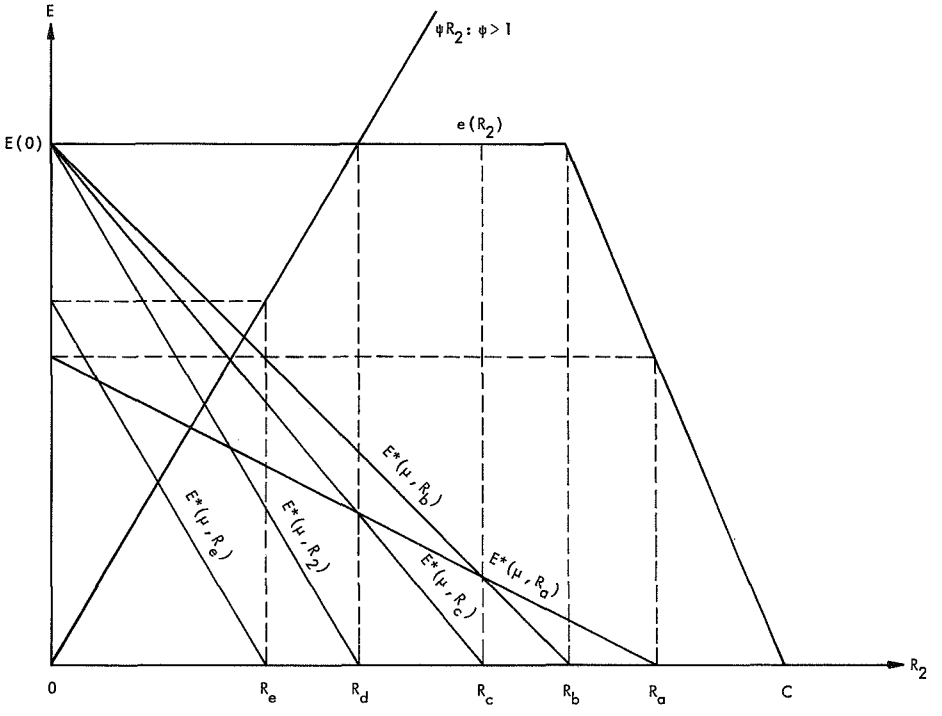

Fig. 19. Construction of $E^{BT*}(R)$ from $e(R_2)$.

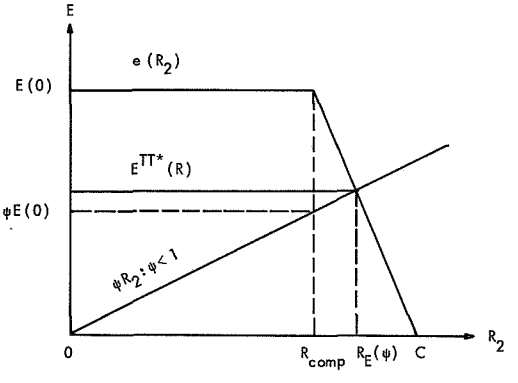

Fig. 20. Construction of $E^{TT*}(R)$ from $e(R_2)$.

composite constraint length of the n-stage code, expressed in channel symbols, is $\left(\prod_{i=1}^{n-1} \nu_i b_i\right)(\psi+1)^{n-1} = \left(\prod_{i=1}^{n-1} \nu_i b_i\right)[n/(n-1)]^{n-1}$, so that the n-stage cascaded tree-code exponent is

$$E^{T^n}(R) = \left(\frac{n-1}{n}\right)^{n-1} \cdot E^{TT^*}(R)\Big|_{\psi=\frac{1}{n-1}} \tag{133}$$

Since $E^{TT^*}(R) \geq \psi e(R) = \frac{1}{n-1}\, e(R)$ for all R, $0 \leq R < C$, and $\left(\frac{n}{n-1}\right)^{n-1} < \left(\frac{n+1}{n}\right)^{n} < \lim_{n\to\infty} \left(\frac{n}{n-1}\right)^{n-1} = \epsilon$, where ϵ is the natural logarithmic base, then the n-stage cascaded tree-code efficiency is bounded by

$$\eta_n(R) \geq \frac{1}{(n-1)\epsilon}, \qquad 0 \leq R < C. \tag{134}$$

This again contrasts sharply with Forney's[10] results for the attainable efficiency in cascading block codes.

5.3.2 Computational Efficiency

One should not infer from the foregoing discussion that cascading is efficient in the sense that it greatly reduces the complexity required to attain a given decoding error probability at a given rate. The actual performance of cascading may be quite contrary in this respect. For a two-stage concatenationlike cascaded coding system in which both constituent codes are tree codes, a maximum-likelihood decoder for the outer code may be required to compare

$$N_o = q_1^{\nu_1 t_1} = \exp \nu_1 t_1 \nu_2 t_2 \psi \ln q$$

$$= \exp \nu_1 \nu_2 b_1 b_2 \psi R \tag{135}$$

alternatives per received branch to achieve a decoding error probability per symbol bounded by

$$P(\epsilon) < \exp -\nu_1 \nu_2 b_1 b_2 \left[E^{TT^*}(R) - o(\nu_1 b_1)\right]. \tag{136}$$

From the discussion between (129) and (130), we see that $E^{TT^*}(R) = e(R) - o(\nu_2 b_2)$ for $\psi \geq 1$. The construction of $E^{TT^*}(R)$ for $\psi \leq 1$ is illustrated in Fig. 20. For every branch of b_1 q_1-ary symbols supplied to the outer decoder, the inner decoder processes $\psi b_1 \nu_2$ received channel symbol branches, and thus makes

$$N_i = \psi b_1 \nu_2 q^{\nu_2 t_2} = \psi b_1 \nu_2 \exp \nu_2 b_2 R_2 \tag{137}$$

calculations. The total number of comparisons made by both decoders per branch supplied to the outer decoder is

$$N = N_o + N_i$$

$$= \left\{1 + \exp \nu_2 b_2 \left[R_2 + \frac{\ln (\nu_2 b_1 \psi)}{\nu_2 b_2} - \nu_1 b_1 \psi R\right]\right\}$$

$$\exp \nu_1 \nu_2 b_1 b_2 \psi R$$

$$= \exp \nu_1 \nu_2 b_1 b_2 \psi R[1-o(\nu_1 b_1)], \tag{138}$$

so that the number of comparisons made by the outer decoder dominates for large outer code constraint lengths and fixed ψ.

We can express $P(\epsilon)$ in terms of the complexity N:

$$P(\epsilon) < N^{-\left\{\frac{E^{TT*}(R)}{\psi R[1-o(\nu_1 b_1)]}\right\}} \tag{139}$$

We compare (139) with the following expression that relates $P(\epsilon)$ to the complexity N_s required for a single-stage maximum-likelihood tree code decoder.

$$P(\epsilon) < N_s^{-\left\{\frac{[e(R)-o(\nu)]}{R}\right\}}. \tag{140}$$

Since $E^{TT*}(R) = [e(R)-o(\nu_2 b_2)]$ for $\psi > 1$, (139) indicates that a concatenationlike cascaded tree code for which $\psi > 1$ requires more complexity than a single-stage tree code to achieve a given error probability at a given rate, if maximum-likelihood decoding is used at each stage. Cascading can therefore offer an improvement only if $\psi < 1$. We observe in both (139) and (140) that the exponents $E^{TT*}(R)/\psi R$ and $e(R)/R$ are monotonic decreasing functions of R. In (140) we observe that for a single-stage code,

$$P(\epsilon) \lesssim \left(\frac{1}{N_s}\right)^{\frac{R_{comp}}{R}}, \qquad 0 < R \leq R_{comp}. \tag{141}$$

On the other hand, since $E^{TT*}(R) \cong \psi R_E(\psi)$ for $0 \leq R \leq R_E(\psi)$ as shown in Fig. 20, (141) becomes

$$P(\epsilon) \lesssim \left(\frac{1}{N}\right)^{\frac{R_E(\psi)}{R}}, \qquad 0 < R \leq R_E(\psi). \tag{142}$$

By choosing ψ sufficiently small, $R_E(\psi)$ becomes arbitrarily close to channel capacity.

We shall now show that essentially the same result can be obtained in a somewhat cleaner form for productlike cascaded tree codes. Our result will be summarized in Theorem 5 following the derivation for productlike codes.

We observe that the smallest attainable value of m in (111) is m = 1, which represents a limiting value of ψ for a given $\nu_2 t_2$. This observation suggests that an expression like (142) applies to productlike codes, with $R_E(\psi)$ replaced by a quantity arbitrarily close to capacity. The composite decoding complexity corresponding to (138) for a productlike cascaded tree code is

$$\begin{aligned} N &= N_o + N_i \\ &= \exp(\nu_1 t_1 \ln q) + \frac{b_1}{t_2} \exp(\nu_2 t_2 \ln q) \\ &= \exp \nu_1 t_1[\ln q - o(\nu_1 t_1)], \qquad \nu_1 t_1 \gg \nu_2 t_2. \end{aligned} \tag{143}$$

Suppose that for some $\epsilon > 0$, the inner code rate is greater than $(1-\epsilon)C$. Furthermore, if (7) is used to bound the error probability p' of the inner decoder, suppose that $\nu_2 t_2$ is sufficiently large to make p' small enough so that the $R'_{comp} = E(0')$ seen by the outer code exceeds $(1-\epsilon) \ln q$, where R'_{comp} can be obtained by substituting (138) and p' in (4) and evaluating at $\rho = 1$. Then, by using (8), the e(R') curve seen by the outer code can be approximated by $(1-\epsilon) \ln q$ for all R', $0 \leq R' \leq (1-\epsilon) \ln q$. Thus for $R' \leq (1-\epsilon)\ln q$,

$$P(\epsilon) < \exp - \nu_1 b_1[(1-\epsilon) \ln q - o(\nu_1 b_1)]. \tag{144}$$

Combining (143) and (144), we express $P(\epsilon)$ in terms of N for a productlike cascaded tree code:

$$\begin{aligned} P(\epsilon) &< N^{-\left[\frac{b_1}{t_1}(1-\epsilon) - o(\nu_2 b_2) - o(\nu_1 b_1)\right]} \\ &= N^{-\left[\frac{C(1-\epsilon)^2}{R} - o(\nu_2 b_2) - o(\nu_1 b_1)\right]} \qquad 0 < R < C(1-\epsilon)^2. \end{aligned} \tag{145}$$

Since ϵ can be made arbitrarily small, we establish the following theorem.

Theorem 5

Consider a two-stage productlike cascaded tree code in which a maximum-likelihood decoder is used at each stage of decoding. Let N be the average number of calculations

per decoded symbol made by the composite decoder. Then for any $\delta > 0$ it is possible to find such a productlike code for which

$$P(\epsilon) \lesssim \{N[1+o(N)]\}^{[(1-\delta)C/R]}, \qquad 0 < R < (1-\delta)C. \tag{146}$$

Since δ is arbitrary, (146) states that we can achieve a decoding error probability that is arbitrarily close to $N^{-(C/R)}$, provided N is sufficiently large.

Heuristically, we can achieve (146) by using an inner code whose rate, R_2, is perhaps $\left(1 - \frac{\delta}{2}\right) C$. The constraint length of the inner code would be chosen sufficiently large to reduce the decoding error probability to the point where the normalized R_{comp} seen by the outer code equals $1 - \frac{\delta}{2}$. This specifies the decoding complexity of the inner code, which may, of course, be very large. The outer code may be used at any normalized rate not exceeding $1 - \frac{\delta}{2}$. When its constraint length is sufficiently large that the decoding complexity of the outer decoder dominates the composite decoding complexity, (146) will then be satisfied.

Theorems 4 and 5 appear to be somewhat paradoxical. Theorem 4 indicates that tree-code cascading is efficient in terms of the coding theorem if a concatenationlike coding system is used. For $\psi = 1$, the coding-theorem efficiency is exactly 1/2 at all rates below capacity. It is tempting to think that a concatenationlike code is necessary to achieve nonzero coding-theorem efficiency. The composite constraint length of a two-stage concatenationlike code is $\alpha = (\psi+1)(\nu_1+1)b_1\nu_2 b_2$, while that of a two-stage product code is $\alpha = (\nu_1+1)b_1\ (\nu_2+1)b_2$. The decoding error probability for the outer decoder is upper-bounded by

$$P(\epsilon) < \exp -\nu_1 b_1[E_1(R')-o(\nu_1 b_1)], \tag{147}$$

where R' is the rate of the outer code. If the code is productlike, $E_1(R') \leq \ln q$ so that, expressed in terms of the composite constraint length in channel symbols,

$$P(\epsilon) < \exp -\alpha[E_p(R')-o(\alpha)], \tag{148}$$

where

$$E_p(R) \leq \frac{\ln q}{(\nu_2+1)b_2}. \tag{149}$$

Thus, as $\nu_2 b_2$ becomes arbitrarily large, $E_p(R)$ becomes arbitrarily small. This indicates that the efficiency of a productlike code cannot be lower-bounded by a nonzero quantity at any rate below capacity. The only way to guarantee that a fixed efficiency can be maintained is to have $E_1(R')$ in (147) grow in proportion to $\nu_2 b_2$, and this can be accomplished through concatenation by making the outer-code alphabet comprise $\psi\nu_2 b_2 k$ symbols (where $k = t_2/b_2$), as in (111), where ψ and k are fixed quantities.

On the other hand, the discussion has indicated that for concatenationlike codes, the asymptotic computational effort required to attain a given, extremely small decoding-error probability at a given composite rate is a monotonically decreasing function of ψ. The computational effort required for productlike codes is equal to the computational effort required for concatenationlike codes in the limiting case as $\psi \to 0$. This result leads to an apparent paradox: Concatenationlike codes have been shown to be efficient in terms of the coding theorem, while there is no guarantee that productlike codes can attain a nonzero efficiency at any rate. On the other hand, productlike codes are computationally more efficient than concatenationlike codes with fixed, nonzero ψ.

We can now further elaborate this paradox. The derivation leading to Theorem 4 showed that the coding-theorem efficiency of a concatenationlike cascaded tree code is exactly 1/2 at all rates below capacity whenever $\psi = 1$. Furthermore, this value of ψ is optimal, in the sense that the efficiency at some rate is less than 1/2 for any other value of ψ. For $\psi > 1$ the efficiency is strictly less than 1/2 at all rates below capacity. For $\psi < 1$, the efficiency is less than 1/2 at very low rates.

An improvement in the efficiency $\eta^{tt}(R)$ occurs, however, for $\psi < 1$ at rates exceeding $R_E(\psi)$. From Fig. 20, the error probability for a two-stage concatenationlike code in which both codes are tree codes is

$$P(\epsilon) < \exp -\nu_1 b_1 \nu_2 b_2 [e(R) - o(\nu_2 b_2) - o(\nu_1 b_1)]$$

$$= \exp -\nu_1 b_1 \nu_2 b_2 (\psi+1) \left[\frac{e(R)}{\psi + 1} - o(\nu_2 b_2) - o(\nu_1 b_1) \right], \tag{150}$$

whenever $R \geq R_E(\psi)$ and $\psi < 1$. Thus

$$\eta^{TT}(R) = \frac{1}{\psi + 1}, \qquad \psi < 1, \quad R \geq R_E(\psi). \tag{151}$$

In particular, $\eta^{TT}(R)$ becomes arbitrarily close to 1 at rates approaching capacity if ψ is fixed but arbitrarily small.

Equation 151 indicates that for a fixed ψ, a concatenationlike code can achieve arbitrary coding-theorem efficiency as $\psi \to 0$. The result is suggestive of the possibility that the coding-theorem efficiency for productlike codes is

$$\eta_p^{TT}(R) \to 1, \qquad R \to C. \tag{152}$$

We shall presently disprove that possibility. The derivation leading to (149) shows that we cannot guarantee a nonzero efficiency for productlike codes. Now we shall present an upper bound to the efficiency attainable by a productlike code. This bound will be derived in terms of the lower bound to tree-code error probability presented by Viterbi.[17] In particular, we shall show that the error exponent in the lower bound for a productlike code grows in proportion to $\nu_2 b_2$, and this will be used to form an upper bound to

the efficiency that may be possible with productlike codes. Then we shall show that this upper bound is arbitrarily small as $R \to C$, so that (152) is not satisfied for productlike codes.

Viterbi[17] showed that the decoding-error probability of a tree code is lower-bounded by

$$P(\epsilon) > \exp -\nu b[e_L(R) + o(\nu b)], \tag{153}$$

where $e_L(R)$ is given parametrically by

$$e_L(R) = E_o(\rho), \qquad 0 < \rho < \infty \tag{154}$$

and

$$R = \frac{E_o(\rho)}{\rho}, \tag{155}$$

where $E_o(\rho)$ is given by (45). Observe that $e_L(R) = e(R)$ for $R_{comp} \leq R < C$.

We want to find a lower bound for the error probability of a productlike code that can be expressed in the form

$$P(\epsilon) > \exp -\nu_1 b_1 \nu_2 b_2 \left[e_L^*(R) + o(\nu_1 b_1) + o(\nu_2 b_2) \right]. \tag{156}$$

This can be used to form an upper bound on the efficiency attainable with productlike tree-code cascading. For simplicity, we consider the binary case, noting that our results carry over to channels with larger alphabets.

Consider a two-stage productlike binary code in which both codes are tree codes. Let the rate of the inner code be R_2 and let the normalized rate of the outer code be $1-\mu$, so that the composite rate is

$$R' = (1-\mu)R_2. \tag{157}$$

The error probability from the inner decoder is bounded by

$$p < \exp -\nu_2 b_2 [e(R_2) - o(\nu_2 b_2)]. \tag{158}$$

Thus the $E_o(\rho)$ seen by the outer code is expressed, using (45), by

$$E_o(\rho) = \rho \ln 2 - (1+\rho) \ln \left[(1-p)^{1/(1+\rho)} + p^{1/(1+\rho)} \right]. \tag{159}$$

As p becomes small by increasing $\nu_2 b_2$, (159) can be approximated by

$$E_o(\rho) \cong \rho \ln 2 - (1+\rho) \ln \left[1 - \frac{\exp\{-\nu_2 b_2 [e(R_2)-o(\nu_2 b_2)]\}}{1+\rho} + \exp\left\{ - \frac{\nu_2 b_2}{1+\rho} [e(R_2)-o(\nu_2 b_2)] \right\} \right]$$

$$\cong \rho \ln 2 - (1+\rho) \exp\left\{ - \frac{\nu_2 b_2}{1+\rho} [e(R_2)-o(\nu_2 b_2)] \right\} \tag{160}$$

Now we can express ρ implicitly in terms of the outer-code rate:

$$R_1 = (1-\mu) \ln 2 = \frac{E_o(\rho)}{\rho}$$

$$\cong \ln 2 - \frac{(1+\rho)}{\rho} \exp\left\{ - \frac{\nu_2 b_2}{1+\rho} [e(R_2)-o(\nu_2 b_2)] \right\} \tag{161}$$

Let K be defined by

$$\mu = \exp -K. \tag{162}$$

Then (161) and (162) imply that

$$K = \frac{\nu_2 b_2}{1+\rho} [e(R_2)-o(\nu_2 b_2)] - \ln \frac{(1+\rho)}{\rho \ln 2}. \tag{163}$$

For large $\nu_2 b_2$ and fixed K, therefore,

$$\rho < \frac{\nu_2 b_2}{K} [e(R_2)-o(\nu_2 b_2)]. \tag{164}$$

Using (161), (162), and (164) in (154), we find that

$$e_L(R') < \nu_2 b_2 \frac{(1-\mu)}{-\ln \mu} (\ln 2)[e(R_2)-o(\nu_2 b_2)]. \tag{165}$$

We wish to maximize (165) subject to (157), and we shall use the notation of (156) to define

$$e_L^*(R) = \sup_{\substack{R'=(1-\mu)R_2 \\ 0<\mu<1}} (\ln 2) \frac{(1-\mu)}{-\ln \mu} e(R_2). \tag{166}$$

We could construct $e_L^*(R)$ graphically from $e(R_2)$ in a similar manner to that which we used in Fig. 19 to construct $E^{BT*}(R)$ from $e(R_2)$. This procedure would provide little insight, however, into the asymptotic behavior of $e_L^*(R)$, especially at rates approaching

capacity. It is simpler, perhaps, to rather crudely bound $e_L^*(R)$. Let ϵ be defined by $R = (1-\epsilon)C$. In the maximization of (166), we are constrained by $R \leq R_2$ and now by $\mu < \epsilon$. Furthermore, the function

$$f(\mu) = \frac{1 - \mu}{-\ln \mu} \tag{167}$$

is a monotonic increasing function of μ, $0 < \mu < 1$. Thus, clearly,

$$e_L^*(R) < \frac{(1-\epsilon)}{-\ln \epsilon} (\ln 2)\, e(R). \tag{168}$$

As $R \to C$, $\epsilon \to 0$, and the ratio of $e_L^*(R)$ to $e(R)$ in (168) becomes arbitrarily small.

The efficiency of a productlike code can now be upper bounded:

$$\eta_p^{TT}(R) < \frac{(1-\epsilon) \ln 2}{-\ln \epsilon} = \frac{R \ln 2}{-C \ln [1-(R/C)]}. \tag{169}$$

Our crude upper bound on $\eta_p^{TT}(R)$ establishes the apparent paradox: A productlike code cannot attain any given level of efficiency in terms of the coding theorem at rates sufficiently close to capacity, while a concatenationlike code can attain an efficiency given by (151) at all rates exceeding $R_E(\psi)$, and this efficiency improves as $\psi \to 0$.

5.4 PRACTICALITY OF CASCADING

We now come to the uncomfortable problem of assessing the practicality of tree-code cascading. Certainly the results above suggest that the cascading of tree codes can be efficient in terms of the coding theorem. The practical question is whether cascading can provide superior performance to alternative decoding techniques at a decreased decoding complexity. Unfortunately, we are unable to give an unqualified affirmative answer to this question, given the current level of development of single-stage tree-code decoding techniques. Nevertheless, we can show that there is an area of application, mainly involving fairly noisy channels, in which the use of cascading appears to offer at least marginal improvement.

5.4.1 Estimated Performance

We shall consider first the matter of which decoding techniques are appropriate for decoding the constituent codes of a two-stage cascaded tree code. Then we shall provide an example that illustrates the marginal improvement that might be gained by using a cascaded tree code in contrast to a single-stage code that uses sequential decoding.

At first, we might consider utilizing either Viterbi algorithm decoding or sequential decoding for both the inner code and the outer code. We shall presently demonstrate that it is impractical to use the Viterbi algorithm for decoding the outer code. For rates

above R_{comp}, we show that if the coding theorem bound (7) is expressed as $P(\epsilon) < p_B$, a Viterbi algorithm decoder requires p_B^{-1} data storage registers and p_B^{-1} computations per received branch. For a data-quality error probability such as $p_B = 10^{-6}$, the complexity $p_B^{-1} = 10^6$ may well be prohibitive. For rates below R_{comp}, we show that for a given decoding error probability and a given computational and storage complexity, it is possible to communicate at a higher rate using sequential decoding than at the attainable rate using the Viterbi decoding algorithm. Our results are obtained from Theorems 6 and 7.

Theorem 6

Suppose the Viterbi decoding algorithm described in section 3.2.1 is applied to a tree code operating at a rate $t/b > R_{comp}$ on a binary symmetric channel. Alternatively, suppose the modified Viterbi decoding algorithm described in section 3.2.4 is applied to a high-rate systematic convolutional code operating on a binary symmetric channel. In either case, if the decoding-error probability bound given by the coding theorem is p_B, then the decoder must contain at least p_B^{-1} data-storage registers, and it must make at least p_B^{-1} calculations per received branch.

Proof of Theorem 6: First we consider the high-rate algorithm described in section 3.2.4. For high-rate systematic convolutional codes, Bucher's[32] results can be manipulated to yield

$$P(\epsilon) < p_B = \exp -\nu[e(R)-o(\nu)]. \tag{170}$$

This contrasts with $\exp -\nu be(R)$ given by (7), which is the general tree-code coding theorem bound for nonsystematic tree codes. Let p_B be expressed as

$$p_B = 2^{-E}. \tag{171}$$

Since $e(R) \leq \ln 2$ for binary codes, comparison of (170) and (148) implies that $\nu \geq E$. From section 3.2.4, the number of calculations per branch is at least $2^{\nu+1}$, and the number of data-storage registers containing maximum-likelihood path history and accumulated relative likelihoods must be at least 2^{ν}. Thus

$$2^{\nu+1} > 2^{\nu} \geq 2^{E} = p_B^{-1}, \tag{172}$$

thereby proving Theorem 6 for the high-rate algorithm described in section 3.2.4.

Now consider the Viterbi decoding algorithm described in section 3.2.1. Whenever $R > R_{comp}$, there is a $\rho < 1$ such that

$$e(R) = \rho R = \rho \frac{t}{b} \ln 2. \tag{173}$$

Thus

$$P(\epsilon) < p_B = \exp -\nu b[e(R)-o(\nu)] = \exp -\nu t[\rho \ln 2 - o(\nu t)], \tag{174}$$

where $0 < \rho < 1$ for $R_{comp} < R < C$. The required storage and the number of calculations per branch, from section 3.2.1, is at least $2^{\nu t}$. Thus, for $R \geq R_{comp}$,

$$2^{\nu t} \geq 2^{\nu t \rho} \geq p_B^{-1}, \tag{175}$$

thereby proving Theorem 6 for the Viterbi decoding algorithm described in section 3.2.1.

Theorem 6 indicates that for a maximum-likelihood decoder, it is not possible to achieve a coding-theorem probability of error that exceeds the reciprocal of the decoding complexity for $R > R_{comp}$, while the discussion in section 5.3.2 indicates that this relationship is possible in the asymptotic limit of large complexity whenever $R < R_{comp}$.

Theorem 7

Consider using either the Viterbi decoding algorithm or sequential decoding to achieve a decoding error probability p' with a data-storage capacity of N and a number of calculations per received branch not exceeding N, where $N < (p')^{-1}$. Then one can communicate at a higher rate using sequential decoding than by using the Viterbi decoding algorithm.

Proof of Theorem 7: In this case the rate t/b achievable when using the Viterbi algorithm, from Theorem 6, must be less than R_{comp}. Let $\rho_v > 1$ be so defined that

$$\frac{t}{b} = \frac{R_{comp}}{\rho_v}. \tag{176}$$

Applying (7) and the results of section 3.2.1, we obtain a decoding-error probability bounded by $2^{-\nu t[\rho_v - o(\nu t)]}$, with a complexity $N = 2^{\nu t}$. Thus, defining $\rho_v^* = \rho_v - o(\nu t)$, we have

$$P(\epsilon) < p_B = (N^{-1})^{\rho_v^*} \tag{177}$$

using the Viterbi decoding algorithm. For sequential decoding the probability of a buffer overflow is bounded by an expression like (175). The rate achievable by using sequential decoding is

$$R_{\rho_v^*} = \frac{1}{\rho_v^*} E_o\left(\rho_v^*\right), \tag{178}$$

where $E_o\left(\rho_v^*\right)$ is given by (45). Thus, since

$$E_o\left(\rho_v^*\right) > E_o(1) = R_{comp}, \qquad \rho_v^* > 1 \tag{179}$$

from Gallager,[3] $R_{\rho_v^*} > t/b$, and Theorem 7 is proved.

Now we must determine what sort of decoding is appropriate for the inner code. Certainly, if we wish to communicate at a composite rate exceeding R_{comp}, then the inner code cannot be decoded by using sequential decoding. By using the Viterbi algorithm for decoding an inner code with a sufficiently large constraint length, it is theoretically possible to use sequential decoding on the outer code and to communicate at any rate below channel capacity. (This subject will be treated in section 5.4.2.) The practical question is whether a reasonably short inner code can be used to reduce the error probability sufficiently that sequential decoding may be used effectively on the outer code.

One application in which cascading appears to be somewhat effective is in increasing the rate at which a communication system can operate with a given, small decoding-error probability, with little increase in complexity over a one-stage coding system using sequential decoding. The improvement is especially noticeable for relatively noisy channels as, for example, the binary symmetric channel with an error probability exceeding .05. Example 4 illustrates this application.

Example 4. We compare two coding systems, designed to yield a decoding error probability of 10^{-8} with a storage requirement of approximately 10^4 symbols, at an arbitrary binary channel error probability p. The first system will be a one-stage coding system in which a tree code is decoded by using sequential decoding, and the second system will be a two-stage productlike cascaded tree-code coding system in which the inner code is decoded according to the Viterbi decoding algorithm and the outer code is decoded by using sequential decoding. [We have also considered an example in which the cascaded tree code was concatenationlike, and in which 2^m-ary symbols were supplied to the outer decoder. The productlike system described here outperformed any concatenationlike system in which $m > 1$.] The basis of comparison will be to determine the ratio of the rates attainable by the two systems, allowing an approximate 10% increase in storage for the cascaded system.

We assume that the constraint lengths of the single-stage code and of the outer code in the cascaded system are sufficiently long that the decoding-error probability is negligible compared with the probability of a buffer overflow. Thus the dominant source of sequential decoder failure is a buffer overflow, whose probability is given by (46). For our example, $P(\epsilon) = 10^{-8}$ and $N = 10^4$, so that (46) implies that $\rho = 2$. We further assume that in the cascaded system, the interleaver causes the effective channel seen by the sequential to be accurately modeled by a binary symmetric channel whose error probability is bounded by the approximation (96) given in section 4.2.4.

For the single-stage code, the attainable normalized rate using sequential decoding with $\rho = 2$, from (47) and (4), is

$$R_\rho = R_2 = 1 - \frac{3}{2 \ln 2} \ln \left[(1-p)^{1/3} + p^{1/3}\right]. \tag{180}$$

If the inner code of the cascaded system is characterized by the parameters $(\nu, t, b, 2)$, the normalized rate attainable with the cascaded system is bounded by

$$R_c > \frac{t}{b} \left\{1 - \frac{3}{2 \ln 2} \ln \left[(1-p_B)^{1/3} + p_B^{1/3}\right]\right\}, \tag{181}$$

where the decoding error probability of the inner code is estimated from (96).

$$P(\epsilon) < p_B = 0.25 \exp -\nu b e\left(\frac{t}{b} \ln 2\right). \tag{182}$$

We can define the rate improvement, I, as the fractional gain in rate obtained by using the cascaded system compared with the rate attainable with the single-stage system:

$$I = \frac{R_c}{R_2} - 1. \tag{183}$$

In Table 8 we have tabulated I as a function of p for the inner codes whose rates and constraint lengths are given. For $p \leq .10$, I_2 has been maximized, subject to the conditions that $\nu t \leq 10$ and ν and t are integers. For $p > .10$, the values obtained for I are believed to be typically good, but it is possible that they could be further increased by adjusting R and ν slightly. Figure 21 shows the improvement factor for our example plotted as a function of p.

Example 4 indicates that for $p \geq .05$, a rate improvement of typically 20% is attainable for a two-stage cascaded tree-code coding system whose complexity is at least 10% greater than that of a single-stage sequential decoding system. It is doubtful whether the improvement obtained by using cascading would be worth the additional cost of its implementation.

We have been unable to discover applications using currently known tree-code decoding techniques in which tree-code cascading offers substantially more improvement over the performance of a single-stage tree code than was accomplished in Example 4. Further accomplishments in developing tree-code decoding techniques will be required before cascading can be made very practical. Several developments could make cascading very effective, as is suggested in section 5.3. One development that would offer substantial improvements for cascading would be to discover a class of tree codes like the BCH block codes that could be easily decoded, and in which the short members at least are efficient in terms of the coding theorem. Better yet would be the discovery of efficient codes operating in higher order alphabets that could be easily decoded, like the Reed-Solomon block codes. Finally, cascading could be made effective at high rates if a modification to the basic Viterbi decoding algorithm like the one described in

Table 8. Representative rate improvement for $\rho = 2$.

p	R	ν	$R_2(p)$	$R_2(p')$	I
.02	2/3	5	.483	.551	.141
.03	5/9	2	.428	.486	.135
.04	1/2	10	.385	.438	.138
.05	1/2	10	.349	.429	.229
.06	5/11	2	.319	.376	.179
.07	2/5	5	.290	.348	.200
.08	2/5	5	.266	.331	.245
.09	5/14	2	.247	.306	.239
.10	1/3	10	.225	.280	.244
.11	1/3	10	.205	.258	.258
.12	2/7	5	.193	.242	.228
.13	2/7	5	.179	.222	.240
.14	1/4	10	.166	.213	.283
.15	1/4	10	.157	.194	.236
.16	1/5	10	.143	.175	.224
.20	1/6	10	.104	.135	.298
.25	1/10	10	.0693	.0874	.262
.30	1/17	10	.0404	.0529	.309

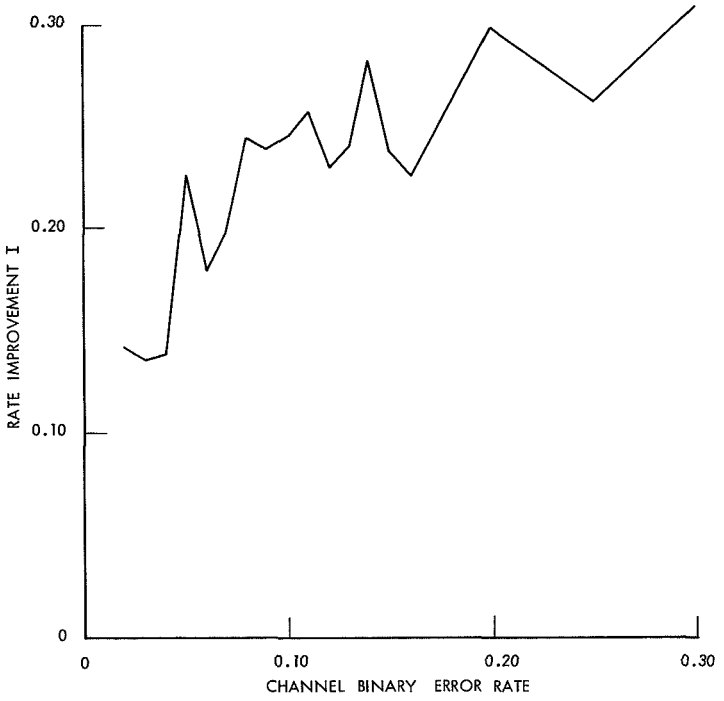

Fig. 21. Rate improvement for Example 4.

section 3.2.4 could be found which would be applicable to efficient nonsystematic high-rate convolutional codes.

5.4.2 Sequential Decoding at Rates Exceeding R_{comp}

Consider now the concatenationlike tree code described in section 5.2.2. We shall show that it is possible, in the asymptotic limit as ν_2 becomes very large, to communicate at any rate below channel capacity by using a Viterbi algorithm decoder for the inner code and by using sequential decoding to decode the outer code, provided the ratio of m to $\nu_2 b_2$ is kept below some bound depending on R, C, and R_{comp}.

At first, it might appear that our assertion contradicts Stiglitz's[30] results. Actually, Stiglitz's results were derived for a two-stage code cascading system in which the inner code is an (n,k) block code with symbols from a q-ary alphabet. For that system, he showed that one could not consider the decoded blocks as symbols from a q^k-ary alphabet, and apply sequential decoding to these q^k-ary symbols at a composite rate exceeding R_{comp}. He did, however, quote Pinsker's[29] result that sequential decoding could be applied at a composite rate exceeding R_{comp} to the individual q-ary symbols obtained from the decoded blocks, provided that they were made to look independent to the decoder. We shall generalize Pinsker's results.

Before proceeding with the derivation, we mention that the required independence of the q^m-ary symbols seen by the outer decoder can be attained by choosing the parameters α and β of the (α,β) interleaver in Fig. 18 sufficiently large. We shall not elaborate here, since this is a problem that is more appropriate for simulation studies than for an analytical solution.

Thus we assume that the derived channel seen by the outer decoder contains statistically independent q^m-ary symbols whose transition probabilities are given by the q^m-ary symmetric channel transition probabilities (119). We recall that Forney[10] showed that the q^m-ary symmetric channel is the worst among all possible q^m-ary channels, in the sense that its error exponent is lowest for a given q^m-ary error probability. Thus if we can prove our assertion for the q^m-ary symmetric channel, which, incidentally, is a reasonably accurate model in our application, then our assertion also applies to any q^m-ary DMC.

Let the inner code operate at a rate R, where $R_{comp} < R < C$. The normalized R^*_{comp} for the 2^m-ary symmetric channel seen by the outer decoder is

$$R^*_{comp} > 1 - \frac{2}{m \ln 2} \ln \left[(1-p_m)^{1/2} + (2^m-1)^{1/2} p_m^{1/2} \right], \tag{184}$$

where $p_m < m \exp -\nu_2 b_2[e(R)-o(\nu_2 b_2)]$. We shall show that if $m/\nu_2 b_2$ is kept below a value depending on R, C, and R_{comp}, then for sufficiently large ν_2, $R^*_{comp} > 1 - \epsilon$ for any $\epsilon > 0$. Then, to communicate at a composite rate $R_c = RR^*_{comp} \geq (1-\delta)C$, for some $\delta > 0$, and still be able to use sequential decoding for decoding the outer code, we can set $R = \left(1 - \frac{\delta}{2}\right)$ and $\epsilon = \frac{\delta}{2}$ so that $R_c = \left(1 - \delta + \frac{\delta^2}{4}\right)C > (1-\delta)C$. This will verify

our original assertion.

For large ν_2 and m, either 2^m or p_m dominates the right-hand expression in brackets in (184). If $2^m p_m$ becomes arbitrarily small, then the entire right-hand term in (184) becomes small, and $R^*_{comp} > 1 - \epsilon$ for some $\epsilon > 0$. The condition for $2^m p_m$ to become arbitrarily small is

$$\frac{m}{\nu_2 b_2} < \frac{e(R) - o(\nu_2 b_2)}{\ln 2}. \tag{185}$$

Thus, if (185) is satisfied, it is possible to make R^*_{comp} approach 1 arbitrarily closely by sufficiently increasing $\nu_2 b_2$, and hence to enable the use of sequential decoding on the outer stage of coding at any composite rate below capacity.

VI. REALIZATION OF OPTIMUM INTERLEAVERS

6.1 INTRODUCTORY CONCEPTS

An interleaver is a device that rearranges the ordering of a sequence of symbols in some one-to-one, deterministic manner. Associated with any interleaver is an unscrambler, which is the device that restores the reordered sequence to its original ordering. Interleavers and unscramblers have a variety of applications in cryptography and in communication technology.

In sections 5.1 and 5.2 interleavers are employed in the generation of several classes of cascaded codes. In the applications dealing with block codes, it is somewhat natural to consider using block interleavers. An example of such a block interleaving function is to divide symbol sequences into blocks corresponding to a two-dimensional array, and to conceptually read the symbols in by rows and out by columns.

In the applications dealing with convolutional codes, such as those given in section 5.2, it is more natural to consider synchronous interleavers, in which a symbol is read out each time a symbol is read in. Synchronous interleavers are a more general class of interleavers than block interleavers, since any block interleaving function can be realized by a synchronous interleaver.

Interleavers and unscramblers are characterized by their encoding delay D, which is the maximum delay encountered by any symbol before it is inserted into the output sequence, and by the storage capacities S and S_u, which are the number of symbols stored by the interleaver and by the unscrambler, respectively.

We define the class of (n_2, n_1) interleavers to be those interleavers that reorder a sequence so that no contiguous sequence of n_2 symbols in the reordered sequence contains any symbols that were separated by fewer than n_1 symbols in the original ordering. We shall present four simple but similar techniques for realizing synchronous (n_2, n_1) interleavers. We shall derive lower bounds for the encoding delay and for the combined storage capacity, $S + S_u$, achievable by any (n_2, n_1) interleaver. An (n_2, n_1) interleaver is optimum if it achieves both the minimum possible encoding delay and the minimum possible combined storage capacity. For any n_1 and n_2 satisfying certain relative primeness conditions, one of the techniques that will be described achieves the minimum possible encoding delay. Then we shall describe reduced-storage versions of these interleavers that also achieve the minimum possible combined storage capacity, and therefore are optimum. Our results are similar to results independently obtained by Forney.[33]

Let $\ldots, a_{z_1}, a_{z_2}, \ldots$ be the sequence of symbols in the output sequence, where $\ldots, z_1, z_2, \ldots$ are the positions of these symbols in the original ordering. For an (n_2, n_1) interleaver, therefore,

$$|z_i - z_j| \geq n_1 \tag{186a}$$

whenever

$$|i-j| \leq n_2 - 1. \tag{186b}$$

Hereafter it will be useful to recognize that the unscrambler for an (n_2, n_1) interleaver is itself an (n_1, n_2) interleaver. This assertion can be verified through the following arguments. Again let $\ldots, a_{z_1}, a_{z_2}, \ldots$ be the sequence of symbols in the output of the interleaver, which is also the input to the unscrambler, where $\ldots, z_1, z_2, \ldots$ are the positions of these symbols in the original ordering. Let $\ldots, z_1', z_2', \ldots$ be the positions of these symbols in the output of the unscrambler. Since the unscrambler restores the sequence to its original ordering,

$$z_i' = z_i + D', \qquad \text{all } i, \tag{187}$$

where D' is a fixed delay introduced by the interleaving-unscrambling process. Thus (186) continues to apply to z_i' and z_j'; that is,

$$|z_i' - z_j'| \geq n_1 \tag{188a}$$

whenever

$$|i-j| \leq n_2 - 1. \tag{188b}$$

But (188) implies that if

$$|z_i' - z_j'| \leq n_1 - 1, \tag{189a}$$

then

$$|i-j| \geq n_2. \tag{189b}$$

This completes the verification, since (189) defines an (n_1, n_2) interleaver.

The encoding delay is defined as

$$D = \sup_j (j - z_j), \tag{190}$$

where $j \geq z_j$ because the interleaver is assumed to be physically realizable. It is assumed that

$$d = \inf_j (j - z_j) = 0, \tag{191}$$

since D could be reduced by d if $d > 0$. It follows that the delay introduced by the combined interleaving and unscrambling operations is also D, which can be seen if

$$z_j' = z_j + D. \tag{192}$$

Finally, if D_u is the encoding delay of the unscrambler, then

$$D_u = \sup_j (z'_j - j) = D. \tag{193}$$

Thus the encoding delays of the interleaver and the unscrambler are both equal to the delay introduced by the over-all interleaving-unscrambling operation.

An (n_2, n_1) interleaver is said to be uniform if the members of every set of n_2 contiguous symbols in the output sequence are mutually separated by at least n_1 symbols in the input sequence, but there is no set of $n_2 + 1$ or more contiguous symbols in the output sequence in which the members are mutually separated by at least n_1 symbols in the input sequence. Clearly, an (n_2, n_1) interleaver is either uniform or nonuniform: If it is nonuniform, then the members of some set of $n_2 + 1$ or more contiguous symbols in the output sequence are mutually separated by at least n_1 symbols in the input sequence.

6.2 FOUR BASIC INTERLEAVING TECHNIQUES

We shall now describe four basic methods of using a commutator and a tapped shift register to realize an (n_2, n_1) interleaver. Subsequently, we shall show that at each point in the (n_2, n_1) plane satisfying a relative primeness condition a modification of one of these methods realizes an optimum (n_2, n_1) interleaver.

6.2.1 Type I (n_2, n_1) Interleaver

Whenever n_1 and $n_2 + 1$ are relatively prime and $n_1 > n_2 + 1$, the device shown in Fig. 22 is a nonuniform (n_2, n_1) interleaver. That device comprises an $[n_2(n_1-1)+1]$-stage shift register with taps at the outermost stages and at every $(n_1-1)^{th}$ intermediate stage, and an (n_2+1)-position commutator that cyclically samples the $n_2 + 1$ taps

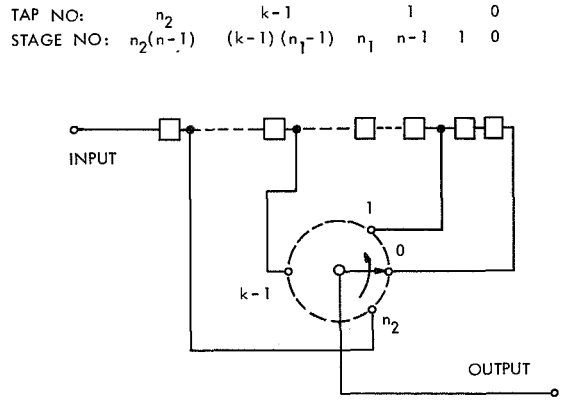

Fig. 22. Type I (n_2, n_1) interleaver.

in reverse order of their distances from the input of the shift register. Observe that the encoding delay of this device is $n_2(n_1-1)$.

Two assertions must be verified to prove that the device is an (n_2, n_1) interleaver: (i) no contiguous sequence of n_2 output symbols contains symbols that were separated by fewer than n_1 symbols in the input sequence; and (ii) each symbol in the input sequence eventually appears in the output sequence. Condition (i) ensures that the device performs the required symbol separation, while condition (ii) is required to show that the device provides a one-to-one mapping of the input sequence into the output sequence.

Assertion (i): Suppose that symbols a_k through $a_{k+n_2(n_1-1)}$ are stored in order in shift-register stages 0 through $n_2(n_1-1)$ when the commutator is at position 0. The device proceeds as follows: Symbol a_k is read out, a new symbol is shifted into the shift register, the commutator is advanced to position 1, symbol a_{k+n_1} is read out, and so on. The ordering of symbols in the output sequence is, therefore, a_k, a_{k+n_1}, a_{k+2n_1}, $\ldots$, a_{k+jn_1}, $\ldots$, $a_{k+n_2n_1}$, a_{k+n_2+1}, $a_{k+n_2+n_1+1}$, $\ldots$, $a_{k+n_2+jn_1+1}$, $\ldots$. We must show that each set of n_2 contiguous output symbols has the required separation. Certainly, each set starting with symbols a_k or a_{k+n_1} has the required separation. Consider now the set of n_2 contiguous symbols in the output sequence starting with a_{k+jn_1} and ending with $a_{k+n_2+(j-2)n_1+1}$, $2 \leq j \leq n_2$. This set can be divided into two subsets, one of which contains symbols a_{k+jn_1} through $a_{k+n_2n_1}$, and the other symbols a_{k+n_2+1} through $a_{k+n_2+(j-2)n_1+1}$. Each subset obviously has the required separation. The lowest index in the first subset is $k+jn_1$, while the highest index in the second subset is $k+n_2+(j-2)n_1+1$. If

$$(k+jn_1) - [k+n_2+(j-2)n_1+1] = 2n_1 - n_2 - 1 \geq n_1, \tag{194}$$

or equivalently if $n_1 \geq n_2+1$, then the entire set has the required separation, since no symbol from one subset was within n_1 symbols of any symbol from the other subset in the original ordering. If $n_1 \leq n_2$, however, the entire set does not have the required separation because the symbol with index $k + n_2 + (j-2)n_1 + 1$ must have been within n_1 symbols of some symbol from the first subset in the original ordering. This completes the proof of Assertion (i).

Assertion (ii): We must show that each input symbol appears somewhere in the output sequence whenever n_1 and $n_2 + 1$ are relatively prime. Let the commutator be at an arbitrary position j when the symbol a_o is first shifted into the shift register. The symbol a_o appears at tap k after $(n_2-k)(n_1-1)$ shifts, and it is read out then if and only if the position of the commutator is also k. But the position of the commutator after $(n_2-k)(n_1-1)$ shifts is $[j+(n_2-k)(n_1-1)] \bmod (n_2+1) = [j-(k+1)(n_1-1)] \bmod (n_2+1)$. Therefore the required condition for a_o to be read out at tap k is

$$[j-(k+1)(n_1-1)-k] \bmod (n_2+1) = [j-(k+1)n_1+1] \bmod (n_2+1) = 0, \tag{195}$$

or equivalently if

$$(kn_1) \bmod(n_2+1) = a, \tag{196}$$

where $a = (j-n_1+1) \bmod(n_2+1)$.

If n_1 and $n_2 + 1$ are relatively prime, then (196) is satisfied for one and only one value of k in the range $0 \leq k \leq n_2$, so that an arbitrary symbol a_o appears once and only once in the output sequence. This establishes Assertion (ii) and verifies that under the given conditions the device shown in Fig. 22 is indeed an (n_2, n_1) interleaver.

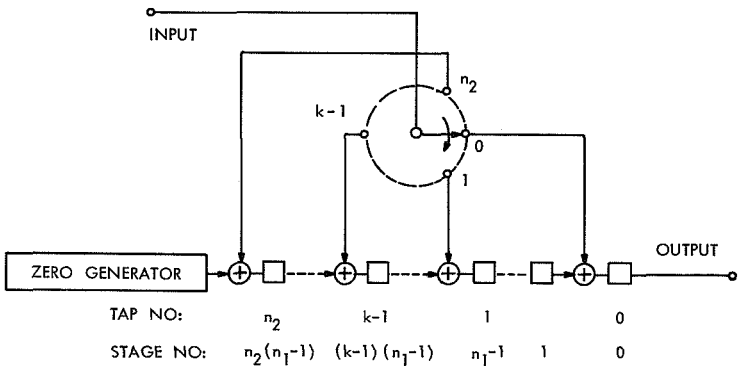

Fig. 23. Unscrambler for the Type I (n_2, n_1) interleaver.

The device shown in Fig. 23 is a simple realization of an unscrambler for the interleaver shown in Fig. 22. By comparing Figs. 22 and 23, the reader may verify that this device restores the original ordering of the sequence of symbols.

6.2.2 Type II (n_2, n_1) Interleaver

Recall that the unscrambling device for an (n_2, n_1) interleaver is an (n_1, n_2) interleaver. Using this fact, we see that whenever n_2 and n_1+1 are relatively prime, and $n_2 > n_1 + 1$, an (n_2, n_1) interleaver can be realized by a device comprising an $[n_1(n_2-1)+1]$-stage shift register with taps at the outermost stages and at every $(n_2-1)^{th}$ intermediate stage, and an (n_1+1)-position commutator that cyclically inserts input symbols into the n_1+1 taps in reverse order of their distances from the input of the shift register. The configuration for this device is shown in Fig. 23 with the parameters n_1 and n_2 interchanged. The corresponding unscrambler can be realized by the device shown in Fig. 22, again with the parameters n_1 and n_2 interchanged.

6.2.3 Type III (n_2, n_1) Interleaver

Whenever n_1 and n_2 are relatively prime, the device shown in Fig. 24 is an (n_2, n_1) interleaver. That device consists of an $[(n_2-1)(n_1+1)+1]$-stage shift register with taps at the outermost stages and at every $(n_1+1)^{th}$ intermediate stage, and an n_2-position

commutator that cyclically samples the n_2 taps in the same order as their distances from the input of the shift register. The encoding delay of this device is therefore $(n_2-1)(n_1+1)$.

A verification that the device shown in Fig. 25 is indeed an (n_2, n_1) interleaver whenever n_1 and n_2 are relatively prime can be given in a manner similar to that given in section 6.2.1 for the Type I interleaver. We shall omit doing so here. Observe that the ordering of symbols in the output sequence is $\ldots, a_k, a_{k-n_1}, a_{k-2n_1}, \ldots, a_{k-(n_2-1)n_1}, a_{k+n_2}, a_{k+n_2-n_1}, \ldots$, so that whenever $n_1 > n_2$, the device shown in Fig. 25 is a uniform (n_2, n_1) interleaver.

A simple realization of an unscrambler for the interleaver of Fig. 24 is given by the device shown in Fig. 25.

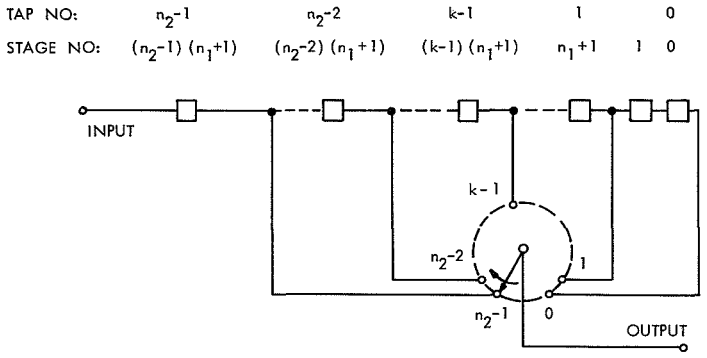

Fig. 24. Type III (n_2, n_1) interleaver.

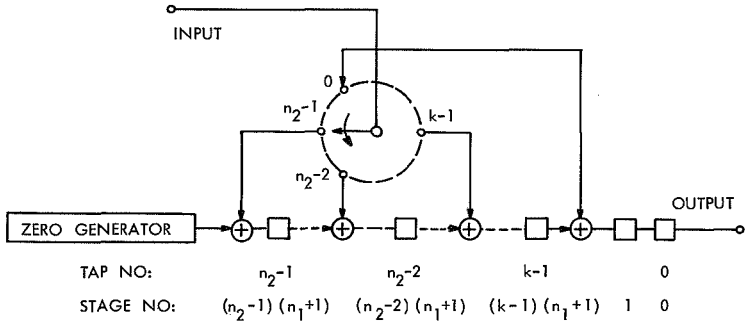

Fig. 25. Unscrambler for the Type III (n_2, n_1) interleaver.

6.2.4 Type IV (n_2, n_1) Interleaver

Whenever n_1 and n_2 are relatively prime, an (n_2, n_1) interleaver can be realized by a device comprising an $[(n_1-1)(n_2+1)+1]$-stage shift register with taps at the outermost

stages and at every $(n_2+1)^{th}$ intermediate stage, and an n_1-position commutator that cyclically inserts input symbols into the n_1 taps in the same order as their distances from the input of the shift register. The configuration for this device is shown in Fig. 25 with the parameters n_1 and n_2 interchanged. The corresponding unscrambler can be realized by the device shown in Fig. 24, with the parameters n_1 and n_2 interchanged.

6.3 OPTIMALITY OF ENCODING DELAY

We shall now show that one of the interleavers of Types I-IV achieves the minimum possible encoding delay for any (n_2, n_1) interleaver, provided the appropriate relative primeness conditions between n_1 and n_2 are satisfied. We first demonstrate that whenever $n_1 > n_2$, either a Type I interleaver or a Type III interleaver achieves the minimum possible encoding delay.

Theorems 8 and 9 are proved in Appendix A.

Theorem 8

The encoding delay for a nonuniform (n_2, n_1) interleaver is at least $n_2(n_1-1)$.

Theorem 9

The encoding delay for a uniform (n_2, n_1) interleaver is at least $(n_2-1)(n_1+1)$.

Theorems 8 and 9 are precise statements of the facts that whenever $n_1 > n_2$, Type I and Type III interleavers achieve the minimum possible encoding delay for nonuniform and uniform (n_2, n_1) interleavers, respectively. Observe that a Type I interleaver provides the minimum possible encoding delay for any (n_2, n_1) interleaver for which $n_2 < n_1 < 2n_2$, and a Type III interleaver provides the minimum possible encoding delay for any (n_2, n_1) interleaver for which $n_1 \geq 2n_2$, provided the appropriate relative primeness conditions on n_1 and n_2 are met.

We shall now demonstrate that whenever $n_1 < n_2$, either a Type II interleaver or a Type IV interleaver achieves the minimum possible encoding delay.

Theorem 10

If an interleaver achieves the minimum possible encoding delay for any (n_2, n_1) interleaver, its unscrambler is an (n_1, n_2) interleaver that achieves the minimum possible encoding delay for any (n_1, n_2) interleaver.

Proof of Theorem 10: Recall that an interleaver and its unscrambler both have the same encoding delay, and that the unscrambler for an (n_2, n_1) interleaver is itself an (n_1, n_2) interleaver. Suppose the theorem were not true. Then the unscrambler for the minimum-delay (n_1, n_2) interleaver would be an (n_2, n_1) interleaver whose delay is less than that of the minimum-delay (n_2, n_1) interleaver. But this is a contradiction, so the theorem must be true.

In conjunction with Theorems 8 and 9, Theorem 10 asserts that a Type II interleaver

provides the minimum possible encoding delay for any (n_2, n_1) interleaver for which $n_1 < n_2 < 2n_1$, and a Type IV interleaver provides the minimum possible encoding delay for any (n_2, n_1) interleaver for which $n_2 \geq 2n_1$, provided the appropriate relative primeness conditions on n_1 and n_2 are met. Thus, for any values of n_1 and n_2 that satisfy the appropriate relative primeness conditions, one of the interleavers of Types I-IV achieves the minimum possible encoding delay.

6.4 REDUCTION AND OPTIMALITY OF STORAGE

Although the basic interleaving techniques achieve the minimum possible encoding delay for any (n_2, n_1) interleaver, they are somewhat wasteful of storage. For example, many of the symbols stored in the later shift-register stages of the Type I or the Type III interleaver have already been read into the output sequence. This fact suggests that it might be possible to reduce the storage capacity of these interleavers without changing their interleaving functions. We shall now describe a technique for efficiently reducing the storage capacity of the basic interleavers, and then demonstrate that these reduced-storage interleavers require the minimum possible combined storage capacity for any (n_2, n_1) interleaver.

We shall examine in some detail techniques for reducing the storage capacity of a Type I interleaver. Consider the symbols that must be stored by the interleaver at any given time. Recall that the ordering of the input symbols in the output sequence is a_k, a_{k+n_1}, a_{k+2n_1}, ..., $a_{k+n_2n_1}$, a_{k+n_2+1}, $a_{k+n_2+1+n_1}$, ..., for some $n_1 > n_2$. We now describe the symbols that must be stored in the interleaver from the time symbol a_k is read out until symbol $a_{k+n_2n_1}$ is read out. From the time symbol a_k is read out of tap 0, the ordering of input symbols read out of tap j is a_{k+jn_1}, $a_{k+jn_1+n_2+1}$, $a_{k+jn_1+2(n_2+1)}$, ..., $0 \leq j \leq n_2$. Thus symbol a_{k+n_1} is the first input symbol that will be read out of tap 1. Let us list all of the input symbols received before symbol a_{k+n_1} that must still be stored by the interleaver. These are the $\left\lfloor \frac{n_1}{n_2+1} \right\rfloor + 1$ symbols, a_k, a_{k+n_2+1}, ..., $a_{k+\ell(n_2+1)}$, for all $\ell(n_2+1) < n_1$, where "$\lfloor x \rfloor$" means "the greatest integer contained in x." All of these symbols will be read out of tap 0. Similarly, symbol a_{k+2n_1} is the first input symbol that will be read out of tap 2. The input symbols that were received before symbol a_{k+2n_1} that must still be stored by the interleaver are the $\lfloor n_2/(n_2+1) \rfloor + 1$ symbols a_{k+n_1}, ..., $a_{k+n_1+\ell(n_2+1)}$, for all $\ell(n_2+1) < n_1$, and the $\lfloor 2n_1/(n_2+1) \rfloor + 1$ symbols a_k, ..., $a_{k+\ell(n_2+1)}$, for all $\ell(n_2+1) < 2n_1$. The first $\lfloor n_1/(n_2+1) \rfloor + 1$ of these symbols will be read out of tap 0, as we have just seen, and then the remainder of these symbols will alternatively be read out of taps 1 and 0. This listing can be continued in an obvious manner. We find that the input symbols that were received before symbol a_{k+jn_1} that must still be

stored by the interleaver include $\lfloor n_1/(n_2+1) \rfloor + 1$ symbols to be read out of tap $j-1$, $\lfloor 2n_1/(n_2+1) \rfloor + 1$ symbols to be read out of tap $j-2$, and so on down to $\lfloor jn_1/(n_2+1) \rfloor + 1$ symbols to be read out of tap 0, $1 \leq j \leq n_2$. The total amount of storage capacity required before symbol a_{k+jn_1} is read out is not quite the sum of these quantities, however, since symbol $a_{k+\ell n_1}$ may be discarded from storage after it has been read out. The total required storage capacity for the interleaver is therefore

$$S = 1 + \sum_{k=1}^{n_2} \lfloor kn_1/(n_2+1) \rfloor. \tag{197}$$

But

$$\sum_{k=1}^{n_2} \lfloor kn_1/(n_2+1) \rfloor = \sum_{k=1}^{n_2} \lfloor (n_2+1-k)n_1/(n_2+1) \rfloor$$

$$= \sum_{k=1}^{n_2} \lfloor n_1 - kn_1/(n_2+1) \rfloor$$

$$= n_2(n_1-1) - \sum_{k=1}^{n_2} \lfloor kn_1/(n_2+1) \rfloor. \tag{198}$$

The last equality follows from the fact that n_1 and n_2+1 are relatively prime. Therefore (197) becomes $S = \frac{1}{2} n_2(n_1-1)+1$. This represents almost a 50% reduction in storage capacity from that used by the Type I interleaver.

The preceding discussion suggests an algorithm for constructing and using an interleaver with minimum storage capacity whose operation is identical to that of the Type I interleaver. We shall first describe the algorithm and then provide a simple example to illustrate its use.

The interleaver is an $\left\lceil \frac{1}{2} n_2(n_1-1)+1 \right\rceil$-stage shift register with taps at positions 0, $\lfloor n_1/(n_2+1) \rfloor$, ..., $\sum_{j=1}^{k} \lfloor jn_1/(n_2+1) \rfloor$, ..., $\frac{1}{2} n_2(n_1-1)$, where the shift-register stages and the tap positions are labeled in reverse order of their distance from the input. For notational purposes, define $\beta = \lfloor n_1/(n_2+1) \rfloor$. Assume that symbols a_k, a_{k+n_2+1}, ..., $a_{k+\beta(n_2+1)}$, a_{k+n_1}, $a_{k+(\beta+1)(n_2+1)}$, $a_{k+n_1+n_2+1}$, ..., are stored in order in the shift-register stages. The algorithm proceeds as follows.

1. Read out symbol a_k from tap 0; then shift in a new input symbol.

2. Read out symbol a_{k+n_1} from tap 1; then shift in a new input symbol, but shift only the shift-register stages from the input through tap 1.

3. For each j, $0 \leq j \leq n_2$, continue the process in the obvious manner: Read out symbol a_{k+jn_1} from tap j; then shift in a new input symbol, but shift only the shift-register stages from the input through tap j.

4. After symbol $a_{k+n_2n_1}$ has been read out and a new input symbol has been shifted into the last shift-register stage, go back and keep repeating steps 1-4.

Example 5

This algorithm can be more easily understood by means of a simple example. Consider a Type I interleaver for which $n_1 = 7$, $n_2 = 3$. It is evident that the ordering of input symbols in the output sequence is 0, 7, 14, 21, 4, 11, 18, 25, 8, 15, 22, 29, ... Now consider the operation of the interleaver shown in Fig. 26, which was designed

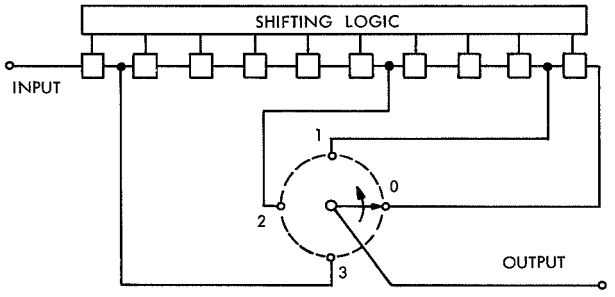

Fig. 26. Interleaver for Example 5.

according to the construction procedure just described. It is a 10-stage shift register with taps at stages 0, 1, 4, and 9. Table 9 lists the symbols stored in the shift-register stages and the symbols that are read out from the initial conditions through the first 5 shifts. The operation of this interleaver is identical to that of the corresponding Type I interleaver, but it requires only 10 shift-register stages instead of 19.

Similar storage-reducing techniques can be applied to the Type II, III, and IV interleavers. The details will not be given here, since they follow closely the methods just described for reducing the storage capacity of Type I interleavers. It turns out that the reduced-storage version of each type of interleaver requires $\frac{1}{2}D + 1$ storage elements, where D is the encoding delay of the basic interleaver. We shall now demonstrate that this realization achieves the minimum possible combined storage capacity for any (n_2, n_1) interleaver and its unscrambler.

Table 9. Steps of interleaver operation for Example 5.

	Symbols Stored in Stages										Symbol Read Out
	9	8	7	6	5	4	3	2	1	0	
Initial Contents	18	16	15	14	12	11	8	7	4	0	0
Shift 1	19	18	16	15	14	12	11	8	7	4	7
Shift 2	20	19	18	16	15	14	12	11	8	4	14
Shift 3	21	20	19	18	16	15	12	11	8	4	21
Shift 4	22	20	19	18	16	15	12	11	8	4	4
Shift 5	23	22	20	19	18	16	15	12	11	8	11

Theorem 11

$$S + S_u \geq D. \tag{199}$$

Proof of Theorem 11: The effect of the combined interleaver-unscrambler operation is to delay the original sequence by D symbols. At the very least, therefore, symbols $a_{i+1}, a_{i+2}, \ldots, a_{i+D}$ must be stored in either the interleaver or the unscrambler when symbol a_i is read out of the unscrambler.

We have shown that at every point in the (n_1, n_2) plane satisfying the appropriate relative primeness conditions one of the four basic interleaver realizations achieves the minimum possible encoding delay. We have then shown that for the reduced-storage versions of these interleavers and unscramblers,

$$S + S_u = D + 2, \tag{200}$$

Both the interleaver and its unscrambler contain one stage of storage more than is absolutely necessary (consider Example 5 with shift-register stage 9 removed), but this initial stage is generally desirable in practical realizations. Except for this

Table 10. Summary of optimum interleaver parameters.

Type	Encoding Delay; also Combined Storage Capacity	Range of Optimality	Restrictions
I	$n_2(n-1)$	$n_2 < n_1 < 2n_2$	n_1, $n_2 + 1$ relatively prime
II	$n_1(n_2-1)$	$n_1 < n_2 < 2n_1$	n_2, $n_1 + 1$ relatively prime
III	$(n_2-1)(n_1+1)$	$n_1 \geq 2n_2$	n_1, n_2 relatively prime
IV	$(n_1-1)(n_2+1)$	$n_2 \geq 2n_1$	n_1, n_2 relatively prime

technicality, the reduced-storage interleavers achieve the minimum possible combined storage requirements for any (n_2, n_1) interleaver and its unscrambler.

We summarize the properties of the optimal interleavers that we have found in Table 10.

APPENDIX A

Proof of Theorems 8 and 9

To prove Theorem 8 we use the following lemma.

Lemma A. 1

Let $a_{z_1}, a_{z_2}, \ldots, a_{z_n}$ be a set of n contiguous symbols in the output sequence of an interleaver, and $z_1, z_2, \ldots, z_n$ be the positions of these symbols in the input sequence. Let $z_i = \max\{z_i, \ldots, z_n\}$ and $z_j = \min\{z_1, \ldots, z_n\}$, $i \neq j$. Then the encoding delay for the interleaver is at least $(z_i - z_j) - (i-j)$.

Proof: From sec. 6. 1, since $D = \sup_j (j-z_j)$ and $0 = \inf_j (j-z_j)$, then

$$D \geq (z_i - z_j) - (i-j) = (j-z_j) - (i-z_i) \geq -D. \tag{A. 1}$$

Proof of Theorem 8: Let $a_{z_1}, a_{z_2}, \ldots, a_{z_{n_2+a}}$ be a set of n_2+a contiguous symbols in the output sequence, and $z_1, z_2, \ldots, z_{n_2+a}$ be the positions of these symbols in the input sequence, where $\{z_1, z_2, \ldots, z_{n_2+a}\}$ are mutually separated by at least n_1. Let $z_i = \max\{z_1, z_2, \ldots, z_{n_2+a}\}$, and $z_j = \min\{z_1, z_2, \ldots, z_{n_2+a}\}$. Since the positions are are mutually separated by at least n_1,

$$z_i - z_j \geq (n_2+a-1)n_1. \tag{A. 2}$$

On the other hand,

$$1 \leq i, \quad j \leq n_2 + a, \qquad i \neq j, \tag{A. 3}$$

so that

$$i - j \leq n_2 + a - 1. \tag{A. 4}$$

Applying Lemma A. 1, we obtain

$$\begin{aligned} D &\geq (z_i - z_j) - (i-j) \\ &\geq (n_2+a-1)n_1 - (n_2+a-1) \\ &= (n_2+a-1)(n_1-1). \end{aligned} \tag{A. 5}$$

For a nonuniform (n_2, n_1) interleaver, $a \geq 1$, and Theorem 8 is proved.

Before proving Theorem 9, we shall first establish some intermediate results. We define a subblock to be the set $\{a_{z_1}, a_{z_2}, \ldots, a_{z_{n_2}}\}$ of n_2 contiguous symbols in the output sequence, where $z_1, z_2, \ldots, z_{n_2}$ denote the positions of these symbols in the input

sequence. The k^{th} adjacent subblock is the set $\left\{a_{z_{(1+kn_2)}}, a_{z_{(2+kn_2)}}, \ldots, a_{z_{(n_2+kn_2)}}\right\}$ of n_2 contiguous symbols in the output sequence. The relative ordering of a subblock is defined to be the ordering of the input positions $z_1, z_2, \ldots, z_{n_2}$.

Lemma A. 2

For a uniform (n_2, n_1) interleaver, the relative ordering of contiguous subblocks is the same.

Proof: From the definition of a uniform (n_2, n_1) interleaver,

$$|z_{k-n_2} - z_k| < n_1. \tag{A. 6}$$

Suppose the relative ordering of two contiguous subblocks differs. Then, for some i and j in the range $1 \leq i \neq j \leq n_2$,

$$z_i \geq z_j + n_1, \tag{A. 7}$$

while

$$z_{j+n_2} \geq z_{i+n_2} + n_1. \tag{A. 8}$$

Suppose $i > j$. Using (A. 8), we obtain

$$\begin{aligned} z_{i+n_2} - z_i &= z_{i+n_2} - z_{j+n_2} + z_{j+n_2} - z_i \\ &\leq -n_1 + \left(z_{j+n_2} - z_i\right). \end{aligned} \tag{A. 9}$$

Since $|z_{i+n_2} - z_i| < n_1$ from (A. 6), (A. 9) implies

$$z_{j+n_2} - z_i > 0. \tag{A. 10}$$

Using (A. 7), however, we have

$$\begin{aligned} z_{j+n_2} - z_j &= z_{j+n_2} - z_i + z_i - z_j \\ &\geq n_1 + \left(z_{j+n_2} - z_i\right). \end{aligned} \tag{A. 11}$$

Since from (A. 6) $|z_{j+n_2} - z_j| < n_1$, (A. 11) implies

$$z_{j+n_2} - z_i < 0. \tag{A. 12}$$

But (A. 10) and (A. 12) are contradictory. A similar contradiction exists when $i < j$. Thus the relative ordering of two contiguous subblocks cannot differ, and the lemma is proved.

Lemma A. 3

Let the boundaries of a subblock of output symbols from a uniform (n_2, n_1) interleaver be chosen so that $z_1 = \min\{z_1, z_2, \ldots, z_{n_2}\}$. Furthermore, let $a = \left(z_{n_2+1} - z_1\right) < n_1$. Then $\max\{z_2, z_3, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + a$.

Proof: Let $z_i = \min\{z_2, z_3, \ldots, z_{n_2}\}$. Since the members of all sets of n_2 contiguous symbols in the output sequence are mutually separated by at least n_1 symbols in the input sequence, $z_i \geq z_1 + a + n_1$, and thus $\max\{z_2, z_3, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + a$.

Lemma A. 4

Let the boundaries of a subblock of output symbols from a uniform (n_2, n_1) interleaver be chosen so that $z_1 = \min\{z_1, z_2, \ldots, z_{n_2}\}$. Furthermore, let $1 \leq a_1 = \left(z_{n_2+1} - z_1\right) < n_1$, and $1 \leq a_2 = \left(z_{2n_2+1} - z_{n_2+1}\right) < n_1$. Then, unless $z_2 > z_3 > \ldots > z_{n_2}$, $\max\{z_2, z_3, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + a_1 + a_2$.

Proof: Suppose that the condition $z_2 > z_3 \ldots > z_{n_2}$ is not satisfied. Then there are two indices j and k such that $j < k$ and $z_j < z_k$. Without loss of generality, suppose that z_j is the ℓ^{th} lowest number of the set $\{z_1, z_2, \ldots, z_{n_2}\}$, and z_k is the $(\ell+1)^{\text{th}}$ lowest number of the set. A simple extension of the proof of Lemma A. 3 establishes that $z_{n_2+j} \geq z_1 + (\ell-1)n_1 + a_1 + a_2$. Then, since $j < k$, $z_k \geq z_{n_2+j} + n_1 = z_1 + \ell n_1 + a_1 + a_2$, so that $\max\{z_2, z_3, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + a_1 + a_2$.

Lemma A. 5

For any subblock of output symbols from a uniform (n_2, n_1) interleaver with finite storage, let $z_i = \min\{z_1, z_2, \ldots, z_{n_2}\}$. Define $a_k = z_{kn_2+i} - z_{(k-1)n_2+i}$. Then

$$\overline{a} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} a_k \approx n_2. \tag{A. 13}$$

Proof: Let $z_j = \max\{z_1, z_2, \ldots, z_{n_2}\}$, and consider the set of input symbols that appears in the first k subblocks of the output sequence. As k becomes arbitrarily large, this set will include all of the symbols a_1 through a_{kn_2}, except for a few symbols near symbols a_1 or a_{kn_2}, under the assumption that the interleaver has finite storage. This assumption also implies that the range $R_k = z_{kn_2+j} - z_{kn_2+i}$ of the k^{th} subblock is also finite. For $0 \leq \ell \leq k-1$, let $z_i^{(L)} = \min_{\ell} z_{\ell n_2+i}$, and let $z_i^{(H)} = \max_{\ell} z_{\ell n_2+i}$. Then

$$\bar{a} = \lim_{N} \frac{1}{N} \sum_{k=1}^{N} a_k = \lim_{k\to\infty} \frac{1}{k} \left(z_i^{(H)} - z_i^{(L)}\right)$$

$$= \lim_{k\to\infty} \frac{1}{k} (kn_2 - R_k) = n_2, \tag{A. 14}$$

and Lemma A. 5 is established.

We have now accumulated enough results to prove Theorem 9.

Proof of Theorem 9: Consider the symbols appearing in a subblock of the output sequence. Without loss of generality, assume that $z_1 = \min\{z_1, z_2, \ldots, z_{n_2}\}$. We consider two cases.

Case I: $z_2 > z_3 > \ldots > z_{n_2}$.

From Lemma A. 5, $\bar{a} = n_2$, and thus for some k

$$a_k = z_{kn_2+1} - z_{(k-1)n_2+1} \geq n_2. \tag{A. 15}$$

Applying Lemma A. 3, (assuming k = 1), we obtain

$$z_2 = \max\{z_1, z_2, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + (n_2-1) + 1. \tag{A. 16}$$

Applying Lemma A. 1, we establish that

$$D \geq (z_2 - z_1) - (2-1)$$

$$\geq (n_2-1)(n_1+1), \tag{A. 17}$$

thereby proving Theorem 9 for Case I.

Case II: The condition $z_2 > z_3 > \ldots > z_{n_2}$ is not satisfied.

From Lemma A. 5, $\bar{a} = n_2$, so that for some k such that $a_k \geq a_{k-1}$,

$$a_{k-1} + a_k \geq 2n_2. \tag{A. 18}$$

Suppose $a_{k-1} < 0$. Then $a_k > 2n_2$, and we can apply Lemma A.3 (assuming k = 1) to obtain

$$\max\{z_1, z_2, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + 2n_2 + 1. \tag{A. 19}$$

Applying Lemma A. 1, and observing that $i - j \leq n_2 - 1$, we obtain

$$D \geq (n_2-1)n_1 + 2n_2 - (n_2-1) + 1 = [(n_2-1)(n_1+1) + 3], \tag{A. 20}$$

thereby proving Theorem 9 for Case II when $a_{k-1} < 0$.

Finally, suppose that $a_{k-1} > 0$. Then we can apply Lemma A.4 (assuming k = 1) to obtain

$$\max \{z_1, z_2, \ldots, z_{n_2}\} \geq z_1 + (n_2-1)n_1 + 2n_2. \tag{A. 21}$$

Applying Lemma A. 1 as before, we obtain

$$D \geq [(n_2-1)(n_1+1)+2], \tag{A. 22}$$

thereby completing the proof of Theorem 9.

Acknowledgment

There are several people and institutions whose assistance has contributed greatly to this work. I wish to express my appreciation to my thesis supervisor, Professor Robert G. Gallager, for his guidance and for his many suggestions which have considerably extended and improved this work. Professor Peter Elias and Professor Chung L. Liu were willing and helpful readers. Dr. Jim K. Omura, a former colleague at Stanford Research Institute, was instrumental in bringing the present topic to my attention. I am also indebted to Dr. Kamil Sh. Zigangirov, of the Institute for Problems of Information Transmission, U. S. S. R., and to Professor Andrew J. Viterbi of the University of California, Los Angeles, for their comments and suggestions.

I am grateful for support from the Research Laboratory of Electronics of Massachusetts Institute of Technology; from the U. S. Government under PL 89-358 (the "G. I. Bill"); and from the Communication Techniques Laboratory, of Stanford Research Institute, Menlo Park, California. I especially wish to thank Dr. Robert F. Daly, of Stanford Research Institute, for his efforts in my behalf.

There are many other people who have given me a great deal of badly needed encouragement. I am especially grateful to Professor Henry J. Zimmermann, of M. I. T., for his advice and for his faith in me throughout my graduate career.

References

1. C. E. Shannon, "A Mathematical Theory of Communication," Bell Syst. Tech. J. 27, 379-423 and 623-656 (1948).
2. C. E. Shannon and W. Weaver, The Mathematical Theory of Communication (University of Illinois Press, Urbana, Ill., 1949).
3. R. G. Gallager, Information Theory and Reliable Communication (John Wiley and Sons, Inc., New York, 1968).
4. R. G. Gallager, "A Simple Derivation of the Coding Theorem and Some Applications," IEEE Trans. on Information Theory, Vol. IT-11, pp. 3-18, 1965.
5. C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, "Lower Bounds to Error Probability for Coding on Discrete Memoryless Channels," Inform. Contr. 10, 65-103 and 522-552 (1967).
6. W. W. Peterson, Error Correcting Codes (The M. I. T. Press, Cambridge, Mass., 1961).
7. E. R. Berlekamp, Algebraic Coding Theory (McGraw-Hill Book Co., New York, 1968).
8. P. Elias, "Error-Free Coding," IRE Trans. on Information Theory, Vol. IT-4, pp. 29-37, 1954.
9. G. D. Forney, Jr., "Concatenated Codes," Technical Report 440, Research Laboratory of Electronics, M. I. T., Cambridge, Mass., December 1, 1965.
10. G. D. Forney, Jr., Concatenated Codes (The M. I. T. Press, Cambridge, Mass., 1966).
11. G. D. Forney, Jr., "Coding System Design for Advanced Solar Missions," Final Report, Contract NAS2-3637, Codex Corporation, Watertown, Mass., 1967.
12. J. K. Omura, "On the Viterbi Decoding Algorithm," IEEE Trans. on Information Theory, Vol. IT-15, No. 1, pp. 177-179, January 1969.
13. J. M. Wozencraft and B. Reiffen, Sequential Decoding (The M. I. T. Press, Cambridge, Mass., 1961).
14. A. D. Wyner and R. B. Ash, "Analysis of Recurrent Codes," IEEE Trans. on Information Theory, Vol. IT-9, pp. 143-156, July 1963.
15. G. D. Forney, Jr., "Convolutional Codes I: Algebraic Structure" (to be published in IEEE Trans. on Information Theory).
16. A. D. Wyner, "On the Equivalence of Two Convolution Code Definitions," IEEE Trans. on Information Theory, Vol. IT-11, No. 4, pp. 600-602, October 1965.
17. A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. on Information Theory, Vol. IT-13, No. 2, pp. 260-269, April 1967.
18. H. L. Yudkin, "Channel State Testing in Information Decoding," Sc. D. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, 1964.
19. J. L. Massey, Threshold Decoding (The M. I. T. Press, Cambridge, Mass., 1963).
20. J. E. Savage, "The Distribution of the Sequential Decoding Computation Time," IEEE Trans. on Information Theory, Vol. IT-12, No. 2, pp. 143-147, April 1966.
21. I. M. Jacobs and E. R. Berlekamp, "A Lower Bound to the Distribution of Computation for Sequential Decoding," IEEE Trans. on Information Theory, Vol. IT-13, No. 2, pp. 167-174, April 1967.
22. F. Jelenik, "An Upper Bound on Moments of Sequential Decoding Effort," IEEE Trans. on Information Theory, Vol. IT-15, No. 1, pp. 140-149, January 1969.
23. F. Jelenik, "A Fast Sequential Decoding Algorithm Utilizing a Stack," IBM J. Res. Develop. 13, 675-685 (1969).

24. K. Sh. Zigangirov, M. S. Pinsker, and B. S. Tsybakov, "Sequential Decoding in a Continuous Channel," Probl. Peredachi Inform., Vol. 3, No. 4, pp. 5-17, 1967.

25. J. L. Massey and M. K. Sain, "Inverses of Linear Sequential Circuits," IEEE Trans. on Computers, Vol. C-17, pp. 330-337, 1968.

26. D. R. Cox and H. D. Miller, The Theory of Stochastic Processes (John Wiley and Sons, Inc., New York, 1965), see Sec. 3.10, pp. 118-123.

27. N. Abramson, "Cascade Decoding of Cyclic Product Codes," IEEE Trans. on Communication Technology, Vol. COM-16, pp. 398-402, 1968.

28. H. O. Burton and E. J. Weldon, Jr., "Cyclic Product Codes," IEEE Trans. on Information Theory, Vol. IT-11, No. 3, pp. 433-439, July 1965.

29. M. S. Pinsker, "On the Complexity of Decoding," Probl. Peredachi Inform., Vol. 1, No. 1, pp. 84-86, 1965.

30. I. G. Stiglitz, "Iterative Sequential Decoding," IEEE Trans. on Information Theory, Vol. IT-15, No. 6, pp. 715-721, November 1969.

31. D. D. Falconer, "A Hybrid Sequential and Algebraic Decoding Scheme," Ph. D. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, 1966.

32. E. A. Bucher, "Error Mechanisms for Convolutional Codes," Technical Report 471, Research Laboratory of Electronics, M. I. T., Cambridge, Mass., August 29, 1969.

33. G. D. Forney, "On the Theory and Practice of Interleaving," IEEE International Symposium on Information Theory, Noordwijk, The Netherlands, June 15-19, 1970.

JOINT SERVICES ELECTRONICS PROGRAM
REPORTS DISTRIBUTION LIST

Department of Defense

Assistant Director (Research)
Office of Director of Defense Research & Engineering
Pentagon, Rm 3C128
Washington, D.C. 20301

Technical Library
DDR&E
Room 3C-122, The Pentagon
Washington, D.C. 20301

Director For Materials Sciences
Advanced Research Projects Agency
Room 3D179, Pentagon
Washington, D.C. 20301

Chief, R&D Division (340)
Defense Communications Agency
Washington, D.C. 20305

Defense Documentation Center
Attn: DDC-TCA
Cameron Station
Alexandria, Virginia 22314

Dr Alvin D. Schnitzler
Institute For Defense Analyses
Science and Technology Division
400 Army-Navy Drive
Arlington, Virginia 22202

Central Intelligence Agency
Attn: CRS/ADD/PUBLICATIONS
Washington, D.C. 20505

M. A. Rothenberg (STEPD-SC(S))
Scientific Director
Deseret Test Center
Bldg 100, Soldiers' Circle
Fort Douglas, Utah 84113

Department of the Air Force

Hq USAF (AFRDDD)
The Pentagon
Washington, D.C. 20330

Hq USAF (AFRDDG)
The Pentagon
Washington, D.C. 20330

Hq USAF (AFRDSD)
The Pentagon
Washington, D.C. 20330
Attn: LTC C. M. Waespy

Colonel E. P. Gaines, Jr.
ACDA/FO
1901 Pennsylvania Avenue N. W.
Washington, D.C. 20451

Dr L. A. Wood, Director
Electronic and Solid State Sciences
Air Force Office of Scientific Research
1400 Wilson Boulevard
Arlington, Virginia 22209

Mr I. R. Mirman
Hq AFSC (SGGP)
Andrews Air Force Base
Washington, D.C. 20331

Rome Air Development Center
Attn: Documents Library (EMTLD)
Griffiss Air Force Base, New York 13440

Mr H. E. Webb, Jr (EMBIS)
Rome Air Development Center
Griffiss Air Force Base, New York 13440

Dr L. M. Hollingsworth
AFCRL (CRN)
L. G. Hanscom Field
Bedford, Massachusetts 01730

Hq ESD (ESTI)
L. G. Hanscom Field
Bedford, Massachusetts 01730

Professor R. E. Fontana, Head
Dept of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson Air Force Base,
Ohio 45433

AFAL (AVT) Dr H. V. Noble, Chief
Electronics Technology Division
Air Force Avionics Laboratory
Wright-Patterson Air Force Base,
Ohio 45433

Director
Air Force Avionics Laboratory
Wright-Patterson Air Force Base,
Ohio 45433

AFAL (AVTA/R. D. Larson)
Wright-Patterson Air Force Base,
Ohio 45433

Director of Faculty Research
Department of the Air Force
U.S. Air Force Academy
Colorado 80840

Academy Library (DFSLB)
USAF Academy, Colorado 80840

Director of Aerospace Mechanics Sciences
Frank J. Seiler Research Laboratory (OAR)
USAF Academy, Colorado 80840

Major Richard J. Gowen
Tenure Associate Professor
Dept of Electrical Engineering
USAF Academy, Colorado 80840

Director, USAF PROJECT RAND
Via: Air Force Liaison Office
The RAND Corporation
Attn: Library D
1700 Main Street
Santa Monica, California 90406

Hq SAMSO (SMTAE/Lt Belate)
Air Force Unit Post Office
Los Angeles, California 90045

AUL3T-9663
Maxwell Air Force Base, Alabama 36112

AFETR Technical Library
(ETV, MU-135)
Patrick Air Force Base, Florida 32925

ADTC (ADBPS-12)
Eglin Air Force Base, Florida 32542

Mr B. R. Locke
Technical Adviser, Requirements
USAF Security Service
Kelly Air Force Base, Texas 78241

Hq AMD (AMR)
Brooks Air Force Base, Texas 78235

USAFSAM (SMKOR)
Brooks Air Force Base, Texas 78235

Commanding General
Attn: STEWS-RE-L, Technical Library
White Sands Missile Range,
New Mexico 88002

Hq AEDC (AETS)
Arnold Air Force Station, Tennessee 37389

European Office of Aerospace Research
Technical Information Office
Box 14, FPO New York 09510

Electromagnetic Compatibility Analysis
Center (ECAC) Attn: ACOAT
North Severn
Annapolis, Maryland 21402

VELA Seismological Center
300 North Washington Street
Alexandria, Virginia 22314

Capt C. E. Baum
AFWL (WLRE)
Kirtland Air Force Base, New Mexico 87117

Department of the Army

Director
Physical & Engineering Sciences Division
3045 Columbia Pike
Arlington, Virginia 22204

Commanding General
U.S. Army Security Agency
Attn: IARD-T
Arlington Hall Station
Arlington, Virginia 22212

Commanding General
U.S. Army Materiel Command
Attn: AMCRD-TP
Washington, D.C. 20315

Director
Advanced Materiel Concepts Agency
Washington, D.C. 20315

Commanding General
USACDC Institute of Land Combat
Attn: Technical Library, Rm 636
2461 Eisenhower Avenue
Alexandria, Virginia 22314

Commanding Officer
Harry Diamond Laboratories
Attn: Dr Berthold Altman (AMXDO-TI)
Connecticut Avenue and
Van Ness Street N. W.
Washington, D.C. 20438

Commanding Officer (AMXRO-BAT)
U.S. Army Ballistic Research Laboratory
Aberdeen Proving Ground
Aberdeen, Maryland 21005

Technical Director
U.S. Army Land Warfare Laboratory
Aberdeen Proving Ground
Aberdeen, Maryland 21005

U.S. Army Munitions Command
Attn: Science & Technology Information Branch, Bldg 59
Picatinny Arsenal, SMUPA-RT-S
Dover, New Jersey 07801

U.S. Army Mobility Equipment Research and Development Center
Attn: Technical Documents Center, Bldg 315
Fort Belvoir, Virginia 22060

Commanding Officer
U.S. Army Engineer Topographic Laboratories
Attn: STINFO Center
Fort Belvoir, Virginia 22060

Dr Herman Robl
Deputy Chief Scientist
U.S. Army Research Office (Durham)
Box CM, Duke Station
Durham, North Carolina 27706

Richard O. Ulsh (CRDARD-IP)
U.S. Army Research Office (Durham)
Box CM, Duke Station
Durham, North Carolina 27706

Technical Director (SMUFA-A2000-107-1)
Frankford Arsenal
Philadelphia, Pennsylvania 19137

Redstone Scientific Information Center
Attn: Chief, Document Section
U.S. Army Missile Command
Redstone Arsenal, Alabama 35809

Commanding General
U.S. Army Missile Command
Attn: AMSMI-RR
Redstone Arsenal, Alabama 35809

Commanding General
U.S. Army Strategic Communications Command
Attn: SCC-CG-SAE
Fort Huachuca, Arizona 85613

Commanding Officer
Army Materials and Mechanics Research Center
Attn: Dr H. Priest
Watertown Arsenal
Watertown, Massachusetts 02172

Commandant
U.S. Army Air Defense School
Attn: Missile Science Division, C&S Dept
P. O. Box 9390
Fort Bliss, Texas 79916

Commandant
U.S. Army Command and General Staff College
Attn: Acquisitions, Lib Div
Fort Leavenworth, Kansas 66027

Mr Norman J. Field, AMCPM-AA-PM
Chief, Program Management Division
Project AACOMS, USAECOM, Bldg. 2525
Fort Monmouth, New Jersey 07703

Mr I. A. Balton, AMSEL-XL-D
Executive Secretary, TAC/JSEP
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703

Commanding General
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703
Attn: AMSEL-SC
DL
GG-DD
XL-D
XL-DT
BL-FM-P
CT-D
CT-R
CT-S
CT-L (Dr W.S. McAfee)
CT-O
CT-I
CT-A
NL-D (Dr H. Bennett)
NL-A
NL-C
NL-P
NL-P-2
NL-R
NL-S
KL-D
KL-I
KL-E
KL-S
KL-SM
KL-T
VL-D
VL-F
WL-D

Dr H. K. Ziegler, Chief Scientist
Army Member TAC/JSEP (AMSEL-SC)
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703

Director (NV-D)
Night Vision Laboratory, USAECOM
Fort Belvoir, Virginia 22060

Commanding Officer
Atmospheric Sciences Laboratory
U.S. Army Electronics Command
White Sands Missile Range,
New Mexico 88002

Commanding Officer (AMSEL-BL-WS-R)
Atmospheric Sciences Laboratory
U.S. Army Electronics Command
White Sands Missile Range,
New Mexico 88002

Chief
Missile Electronic Warfare Tech
Area (AMSEL-WL-M)
Electronic Warfare Laboratory, USAECOM
White Sands Missile Range,
New Mexico 88002

Product Manager NAVCON
Attn: AMCPM-NS-TM, Bldg 439
(H. H. Bahr)
Fort Monmouth, New Jersey 07703

Department of the Navy

Director, Electronics Programs
Attn: Code 427
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

Commander
Naval Security Group Command
Naval Security Group Headquarters
Attn: Technical Library (G43)
3801 Nebraska Avenue, N. W.
Washington, D.C. 20390

Director
Naval Research Laboratory
Washington, D.C. 20390
Attn: Code 2027
Dr W. C. Hall, Code 7000
Mr A. Brodzinsky, Supt,
Electronics Div

Code 8050
Maury Center Library
Naval Research Laboratory
Washington, D.C. 20390

Dr G. M. R. Winkler
Director, Time Service Division
U.S. Naval Observatory
Washington, D.C. 20390

Naval Air Systems Command
AIR 03
Washington, D.C. 20360

Naval Ship Systems Command
Ship 031
Washington, D.C. 20360

Naval Ship Systems Command
Ship 035
Washington, D.C. 20360

U.S. Naval Weapons Laboratory
Dahlgren, Virginia 22448

Naval Electronic Systems Command
ELEX 03, Rm 2534 Main Navy Bldg
Department of the Navy
Washington, D.C. 20360

Commander
U.S. Naval Ordnance Laboratory
Attn: Librarian
White Oak, Maryland 20910

Director
Office of Naval Research
Boston Branch
495 Summer Street
Boston, Massachusetts 02210

Commander (ADL)
Naval Air Development Center
Attn: NADC Library
Johnsville, Warminster,
Pennsylvania 18974

Commander (Code 753)
Naval Weapons Center
Attn: Technical Library
China Lake, California 93555

Commanding Officer
Naval Weapons Center
Corona Laboratories
Attn: Library
Corona, California 91720

Commanding Officer (56322)
Naval Missile Center
Point Mugu, California 93041

W. A. Eberspacher, Associate Head
Systems Integration Division, Code 5340A
U.S. Naval Missile Center
Point Mugu, California 93041

Commander
Naval Electronics Laboratory Center
Attn: Library
San Diego, California 92152

Deputy Director and Chief Scientist
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California 91101

Library (Code 2124)
Technical Report Section
Naval Postgraduate School
Monterey, California 93940

Glen A. Myers (Code 52 Mv)
Assoc Professor of Electrical Engineering
Naval Postgraduate School
Monterey, California 93940

Commanding Officer (Code 2064)
Navy Underwater Sound Laboratory
Fort Trumbull
New London, Connecticut 06320

Commanding Officer
Naval Avionics Facility
Indianapolis, Indiana 46241

Director
Naval Research Laboratory
Attn: Library, Code 2039 (ONRL)
Washington, D.C. 20390

Commanding Officer
Naval Training Device Center
Orlando, Florida 32813

U.S. Naval Oceanographic Office
Attn: M. Rogofsky, Librarian (Code 1640)
Washington, D.C. 20390

Other Government Agencies

Dr H. Harrison, Code RRE
Chief, Electrophysics Branch
National Aeronautics and
Space Administration
Washington, D.C. 20546

NASA Lewis Research Center
Attn: Library
21000 Brookpark Road
Cleveland, Ohio 44135

Los Alamos Scientific Laboratory
Attn: Reports Library
P. O. Box 1663
Los Alamos, New Mexico 87544

Mr M. Zane Thornton, Chief
Network Engineering, Communications
and Operations Branch
Lister Hill National Center for
Biomedical Communications
8600 Rockville Pike
Bethesda, Maryland 20014

U.S. Post Office Department
Library - Room 6012
12th & Pennsylvania Ave. N. W.
Washington, D.C. 20260

Non-Government Agencies

Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Mr Jerome Fox, Research Coordinator
Polytechnic Institute of Brooklyn
333 Jay Street
Brooklyn, New York 11201

Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, New York 10027

Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

Director
Stanford Electronics Laboratory
Stanford University
Stanford, California 94305

Director
Microwave Laboratory
Stanford University
Stanford, California 94305

Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

Director
Electronics Sciences Laboratory
University of Southern California
Los Angeles, California 90007

Director
Electronics Research Center
The University of Texas at Austin
Engineering-Science Bldg 110
Austin, Texas 78712

Division of Engineering and
Applied Physics
210 Pierce Hall
Harvard University
Cambridge, Massachusetts 02138

Dr G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

Dr John C. Hancock, Head
School of Electrical Engineering
Purdue University
Lafayette, Indiana 47907

Dept of Electrical Engineering
Texas Technological University
Lubbock, Texas 79409

Aerospace Corporation
P. O. Box 95085
Attn: Library Acquisitions Group
Los Angeles, California 90045

Airborne Instruments Laboratory
Deerpark, New York 11729

The University of Arizona
Department of Electrical Engineering
Tucson, Arizona 85721

Chairman, Electrical Engineering
Arizona State University
Tempe, Arizona 85281

Engineering and Mathematical
Sciences Library
University of California at Los Angeles
405 Hilgred Avenue
Los Angeles, California 90024

Sciences-Engineering Library
University of California
Santa Barbara, California 93106

Professor Nicholas George
California Institute of Technology
Pasadena, California 91109

Aeronautics Library
Graduate Aeronautical Laboratories
California Institute of Technology
1201 E. California Boulevard
Pasadena, California 91109

Hunt Library
Carnegie-Mellon University
Schenley Park
Pittsburgh, Pennsylvania 15213

Dr A. G. Jordan
Head of Dept of Electrical Engineering
Carnegie-Mellon University
Pittsburg, Pennsylvania 15213

Case Western Reserve University
Engineering Division
University Circle
Cleveland, Ohio 44106

Hollander Associates
Attn: Librarian
P. O. Box 2276
Fullerton, California 92633

Dr Sheldon J. Welles
Electronic Properties Information Center
Mail Station E-175
Hughes Aircraft Company
Culver City, California 90230

Illinois Institute of Technology
Department of Electrical Engineering
Chicago, Illinois 60616

Government Documents Department
University of Iowa Libraries
Iowa City, Iowa 52240

The Johns Hopkins University
Applied Physics Laboratory
Attn: Document Librarian
8621 Georgia Avenue
Silver Spring, Maryland 20910

Lehigh University
Department of Electrical Engineering
Bethlehem, Pennsylvania 18015

Mr E. K. Peterson
Lenkurt Electric Co. Inc.
1105 County Road
San Carlos, California 94070

MIT Lincoln Laboratory
Attn: Library A-082
P. O. Box 73
Lexington, Massachusetts 02173

Miss R. Joyce Harman
Project MAC, Room 810
545 Main Street
Cambridge, Massachusetts 02139

Professor R. H. Rediker
Electrical Engineering, Professor
Massachusetts Institute of Technology
Building 13-3050
Cambridge, Massachusetts 02139

Professor Joseph E. Rowe
Chairman, Dept of Electrical Engineering
The University of Michigan
Ann Arbor, Michigan 48104

New York University
Engineering Library
Bronx, New York 10453

Professor James A. Cadzow
Department of Electrical Engineering
State University of New York at Buffalo
Buffalo, New York 14214

Department of Electrical Engineering
Clippinger Laboratory
Ohio University
Athens, Ohio 45701

Raytheon Company
Research Division Library
28 Seyon Street
Waltham, Massachusetts 02154

Rice University
Department of Electrical Engineering
Houston, Texas 77001

Dr Leo Young, Program Manager
Stanford Research Institute
Menlo Park, California 94025

Sylvania Electronic Systems
Applied Research Laboratory
Attn: Documents Librarian
40 Sylvan Road
Waltham, Massachusetts 02154

Dr W. R. LePage, Chairman
Department of Electrical Engineering
Syracuse University
Syracuse, New York 13210

Dr F. R. Charvat
Union Carbide Corporation
Materials Systems Division
Crystal Products Department
8888 Balboa Avenue
P. O. Box 23017
San Diego, California 92123

Utah State University
Department of Electrical Engineering
Logan, Utah 84321

Research Laboratories for the
Engineering Sciences
School of Engineering and Applied Science
University of Virginia
Charlottesville, Virginia 22903

Department of Engineering and
Applied Science
Yale University
New Haven, Connecticut 06520

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Research Laboratory of Electronics Massachusetts Institute of Technology Cambridge, Massachusetts 02139	Unclassified
	2b. GROUP None

3. REPORT TITLE

Cascaded Tree Codes

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

John L. Ramsey

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1, 1970	120	33

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DA 28-043-AMC-02536(E) b. PROJECT NO. 20061102B31F	Technical Report 478
c. NASA Grant NGL 22-009-013 d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) None

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program Through U. S. Army Electronics Command

13. ABSTRACT

Cascaded codes are long codes that are constructed by successively encoding a series of relatively short constituent codes. The purpose of cascading is to facilitate decoding by dividing the composite decoding process into a sequence of relatively simple steps, each of which corresponds to the decoding of one of the constituent codes.

We study cascading techniques in which the constituent codes are tree codes. We determine the efficiency attainable with cascading, and bound the attainable error probability in terms of the composite decoding complexity. Our major results in these areas are the following.

1. A 2-stage cascaded tree code can be formulated to yield an error exponent that equals 1/2 of the single-stage error exponent at all rates below capacity.

2. If **N** is the composite decoding complexity per decoded symbol for a cascaded tree code in which maximum-likelihood decoding is applied to each constituent code, then in the limit of asymptotically large N one can find a code for which the decoding error probability becomes arbitrarily close to $(1/N)^{(C/R)}$

3. Sequential decoding can be used on the outer stage of a cascaded tree code whose composite rate exceeds R_{comp}, provided the alphabet sizes of the constituent codes are suitably restricted.

We also show how to apply the Viterbi decoding algorithm to an unterminated tree code, and describe the burst characteristics of decoding errors made by a Viterbi decoder. Finally, we present techniques for efficiently realizing a useful class of synchronous interleavers.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Cascaded Decoding						
Coding						
Communication Theory						
Convolutional Codes						
Information Theory						
Interleavers						
Iterated Coding						
Tree Codes						