MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

62 G - 4

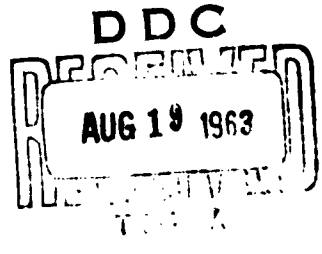
A QUALITATIVE DESCRIPTION OF SEQUENTIAL DECODING

Irwin L. Lebow

12 July 1963

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the joint support of the U. S. Army, Navy and Air Force under Air Force Contract AF 19(628) - 500.

LEXINGTON MASSACHUSETTS



ABSTRACT

This report contains a qualitative discussion of sequential decoding directed toward the non-specialist in information theory. Both the Wozencraft and Fano algorithms are included. The description of sequential decoding is preceded by a summary of the general problem of probabilistic coding and decoding.

Introduction

In 1948 C.E. Shannon stated his so-called coding theorem, that one can communicate at rates arbitrarily close to a maximum called the channel capacity with arbitrarily low error probabilities provided one uses suitably complex signals with which to modulate the channel. Finding the suitably complex signals or codes which are also realizable in practical equipment has been a formidable problem. The particular coding scheme which seems most promising in making Shannon's theorem practical was developed by J. M. Wozencraft[1] in 1958 and is called sequential decoding. A machine (SECO[2]) realizing these principles was constructed at Lincoln Laboratory and when used on a telephone line[3] resulted in a 4-fold increase in the data rate over conventional techniques with a negligibly small error probability. A variation of the original scheme was developed by R. M. Fano[4] and first results[5] obtained in simulating this scheme are promising.

In this report we discuss the general principles of sequential decoding at a level which it is hoped will be understandable to the non-specialist in information theory. Both the Wozencraft and Fano schemes are discussed and an effort is made to point out their basic similarities and their differences.

The description of sequential decoding is preceded by a qualitative discussion of the general coding problem.

Discrete Communications Channels

The term channel is a very overworked one. We speak of the gaussian channel, telephone channel, H-F channel, satellite channel, binary symmetric channel, etc. Some of these words are descriptive of transmission media, some are descriptive of noise perturbations. We shall begin with a definition of a channel in the information theoretic sense and use the word only in this context throughout this note.

A channel is described by first determining a set of signals or waveforms which the transmitter is allowed to use. We can label each of the waveforms with a symbol, say a_i, and we call the set of these symbols or waveforms the input alphabet. These waveforms are transmitted to a receiver through a communications medium which, in general, distorts the signals. The receiver, in turn, operates on the received signal and thereby generates a set of output symbols b_j called the output alphabet. A channel is completely specified if

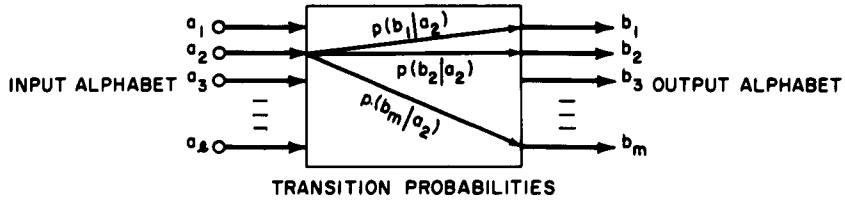


Fig. 1. A discrete channel.

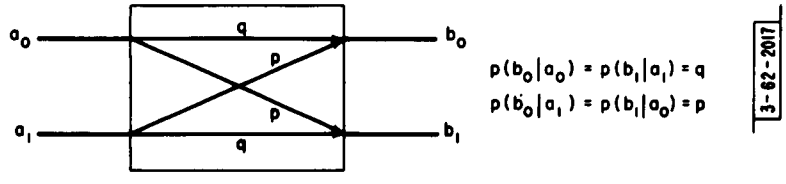


Fig. 2. The binary symmetric channel.

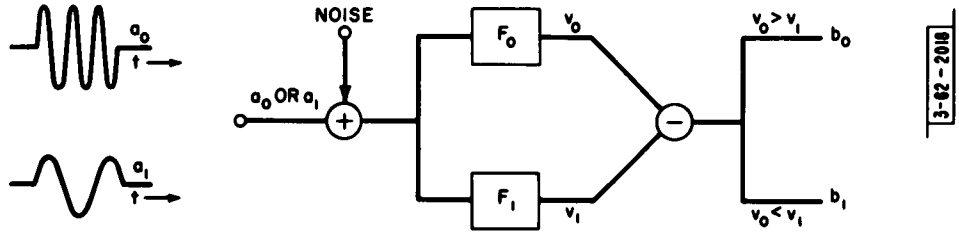


Fig. 3. An approximation to the binary symmetric channel.

the distortion introduced in the transmission can be completely described, in the sense that if symbol a_i is transmitted, we know the probability of receiving each of the b_j's. Thus a channel can be represented by the block diagram of Fig. 1 where we show an input alphabet A, an output alphabet B, and a set of arrows from each a_i to each b_j, each arrow labelled by the conditional probability $p(b_j \mid a_i)$. Thus the channel includes the transmitter and receiver as well as the transmission medium.

The above definition of a channel is strictly true only for a so-called memoryless channel, i.e., a channel in which it is meaningful to speak of $p(b_j \mid a_i)$ without reference to previously transmitted symbols. In this note we shall consider only memoryless channels.

The binary-symmetric channel, shown in Fig. 2, is a simple example of a channel. Here, the transmitter can send one of two signals, a_0 and a_1 and the receiver gives one of two answers b_0 and b_1. The channel transition probabilities are symmetric as shown in the figure. Since both alphabets contain the same number of signals, there is no loss of generality in labelling a_0 and b_0 by the symbol 0 and a_1 and b_1 by the symbol 1, and this is usually done. It will be noticed that this description of the binary-symmetric channel says nothing specific about the physical transmission medium used.

One method of building an approximation to a binary symmetric channel is shown in Fig. 3. The transmitted signals a_0 and a_1 are pulses of one or the other of two carrier frequencies in the H-F region. If fading due to ionospheric effects is neglected the received signal differs from the transmitted signal in that random noise is added at the front end of the receiver. The latter contains two filters each tuned to one of the two transmitted frequencies. Let v_0 and v_1 represent the output voltages of the two filters. The receiver output is defined to be symbol b_0 if $v_0 > v_1$, and b_1 if $v_1 > v_0$. Since $p(b_0 \mid a_1) = p(b_1 \mid a_0) = p$ the channel is binary-symmetric. A real H-F system of this kind will at best be only approximately binary-symmetric since the ionospheric behavior will cause departures of the operating system from the ideal.

In Fig. 4 we add a variation to the channel of Fig. 3. We include a third symbol b_2 in the output alphabet. In this channel b_0 is obtained if $v_0 - v_1 > T$; b_1, if $v_1 - v_0 > T$; and b_2, if $|v_0 - v_1| \leq T$, where T is some positive threshold voltage. Thus this channel can be described by

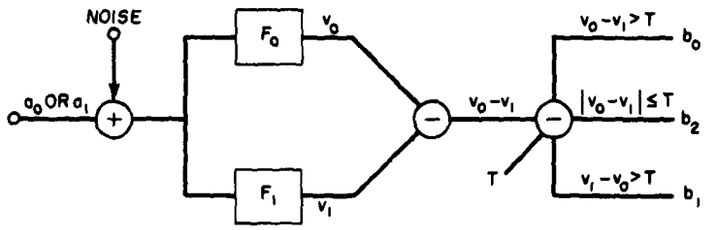


Fig. 4. An approximation to the binary erasure channel.

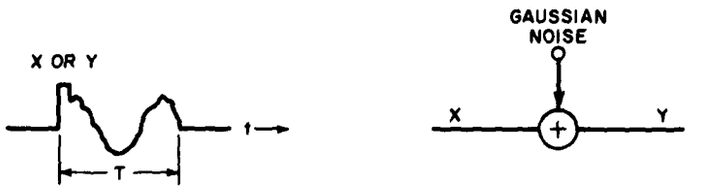


Fig. 5. A Gaussian channel.

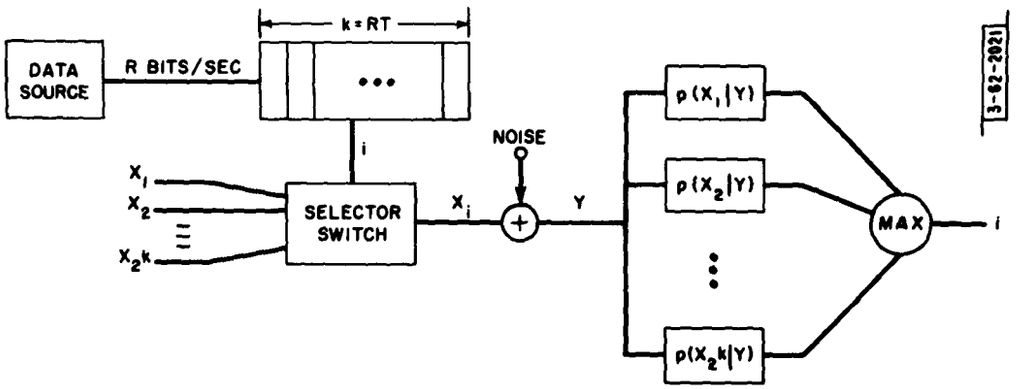


Fig. 6. Coding for the continuous channel.

$$p(b_0|a_1) = p(b_1|a_0) = p_1$$

$$p(b_2|a_1) = p(b_2|a_0) = p_2$$

$$p(b_0|a_0) = p(b_1|a_1) = 1 - p_1 - p_2$$

If $p_1 = 0$, then the channel is the so-called binary-erasure channel, which, though at most an approximation to some real channels, has some convenient mathematical properties.

Continuous Channels

The previous section considered so-called discrete channels; i. e., channels where the input and output alphabets are of finite size. Consider the channel shown in Fig. 5. Here the input alphabet is the set of all time functions in the interval $0 < t < T$ with average power $< S$. The channel perturbation takes the form of white gaussian noise with power density N_0 watts per cycle. The received signal is obtained by adding to the transmitted signal an output sample of the noise generator. This is the so-called white gaussian channel. If the signals and noise are restricted to a bandwidth W, then the channel is the band-limited gaussian channel. Clearly a gaussian channel can be converted into a binary-symmetric channel by selecting two particular time functions for the input alphabet and by defining the output alphabet as in Fig. 3.

The channel of Fig. 5 is an example of a continuous channel. It is useful for our purposes not because it is practical in its own right but because it is a limiting case against which we can compare the more practical discrete channels. It is in terms of this continuous channel that we shall introduce the coding theorem of Shannon.

We code for this continuous channel by selecting $2^k = 2^{RT}$ input waveforms of length T and labelling these waveforms $X_1, X_2 \ldots X_2k$. If symbol X_j is transmitted, the received waveform is $Y = X_j + n$, where n is a noise waveform. The receiver thereupon decodes by computing the numbers $P(X_1|Y), P(X_2|Y), \ldots, P(X_2k|Y)$. That is, given that Y was received the receiver computes the probability that each of the various input symbols was transmitted, based upon its knowledge of the channel The receiver or decoder output is defined in terms of these probabilities: the receiver output is the symbol Y_i if $P(X_i|Y)$ is the largest probability. In other words, the

receiver computes that symbol which was most probably transmitted based upon the evidence furnished by the received waveform Y. If symbol X_j was transmitted and symbol Y_i is received where $i \neq j$, then the receiver makes an error.

The system we have been describing is shown in Fig. 6. In it, a data source emits R bits per second. These data bits are shifted serially into a $k = RT$ cell shift register. The shift register is filled in T seconds. Each of the 2^k values of the register is associated with one of the 2^k input symbols X_j of the code. Hence every T seconds one of the waveforms X_j is transmitted and one of the symbols Y_i is received. Let $P_e(T)$ be the probability of error, that is, the probability that the receiver makes an incorrect decision.

Now suppose we repeat the above process by selecting a different set of 2^k waveforms of length T as the code. We obtain in this way another probability of error. Suppose that we make all possible selections of 2^k waveforms and compute the probability of error for each code.

Let us hold R fixed and repeat this process for a larger value of T or equivalently a larger value of $k = RT$. Shannon's coding theorem states that there exist codes (selections of waveforms) for which $P_e(T)$ tends to 0 as T becomes large, provided that the rate R is less then some maximum value, called the channel capacity C. If $R > C$ then there is no way of making $P_e(T)$ tend to 0 for large T. This behavior of the error probability can be stated more strongly: there exist codes of length T seconds where the probability of error P_e behaves according to

$$P_e \sim 2^{-TE_1(R)}, \quad R < C$$

where the exponent $E_1(R)$ is a function having the general behavior shown in Fig. 7. Here $E_1(R)$ is plotted against the rate R. It has some positive value at $R = 0$ and decreases to 0 when $R = C$, the channel capacity. Since the probability of error depends upon the product E_1T, there is a choice as to how to obtain a given P_e. If one chooses to operate at very low rates with respect to channel capacity, then E_1 is large and T is correspondingly small. For operation at rates which are a large fraction of capacity, E_1 is small and the necessary T becomes correspondingly large. Thus one can obtain an arbitrarily small P_e for a given $R < C$, provided one is willing to make T

large enough. But this implies generating 2^{RT} waveforms at the transmitter and computing the largest of 2^{RT} numbers at the receiver. In general 2^{RT} is too large for practical exploitation of the continuous channel.

Channels with Discrete Input and Continuous Output

The first step in making practical use of the coding theorem is to limit the size of the channel input alphabet to some reasonable number, say $m = 2^{\ell}$, where ℓ is generally less than 5 or 6. In the simplest case $\ell = 1$ and we have a binary input alphabet. Suppose these binary symbols are each τ seconds long. Then in time T we can transmit $n = T/\tau$ binary channel symbols or a total of 2^n different sequences of channel symbols in time T. To code for the channel we select $2^k = 2^{RT}$ of these 2^n sequences. Such a code is often called an (n, k) block code. This is to be contrasted with the selection of 2^k of the infinite number of different waveforms of length T in the continuous channel.

More generally, if the input alphabet size is 2^{ℓ} and each waveform is of length τ, then there are $2^{\ell n}$ possible sequences of length $n = T/\tau$. Obviously, $k < \ell n$ or $k/\ell n < 1$. The fraction $k/\ell n$, the ratio of the number of information bits per block to the number of equivalent binary symbols per block, is often called the rate r in bits per transmitted binary symbol.

When the input alphabet of the channel is restricted in this way the picture of the transmitting terminal presented in Fig. 6 for the continuous channel changes. In the continuous case the channel alphabet consisted of 2^k waveforms one of which was selected for each set of k bits received from the source. For the finite input alphabet (binary case) we have the picture shown in Fig. 8. For every set of k bits received from the source, n bits are generated. Generally one can think of the n bits as being composed of the k information bits to which are appended $n - k$ parity check digits which are linear functions of the k information bits. These n digits are taken serially to select one of the two waveforms forming the channel input alphabet. In the continuous case the encoder is simply a device for selecting one of 2^k waveforms. In the discrete input alphabet case, the encoder first performs a transformation on the k input digits and then makes n waveform selections in sequence.

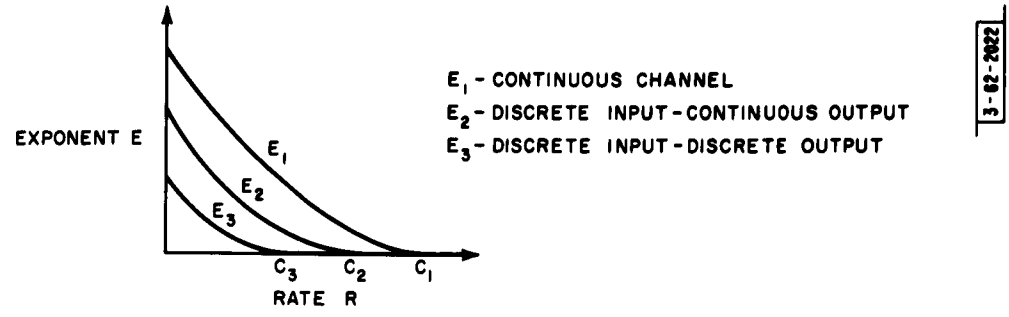


Fig. 7. Error exponents.

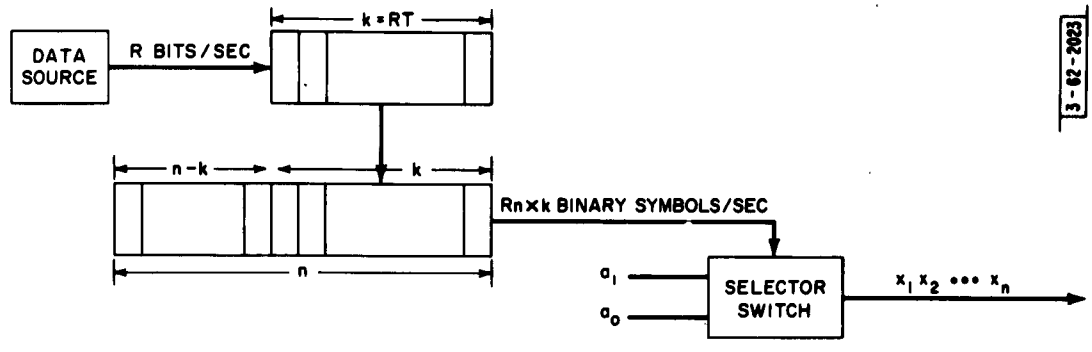


Fig. 8. A binary encoder.

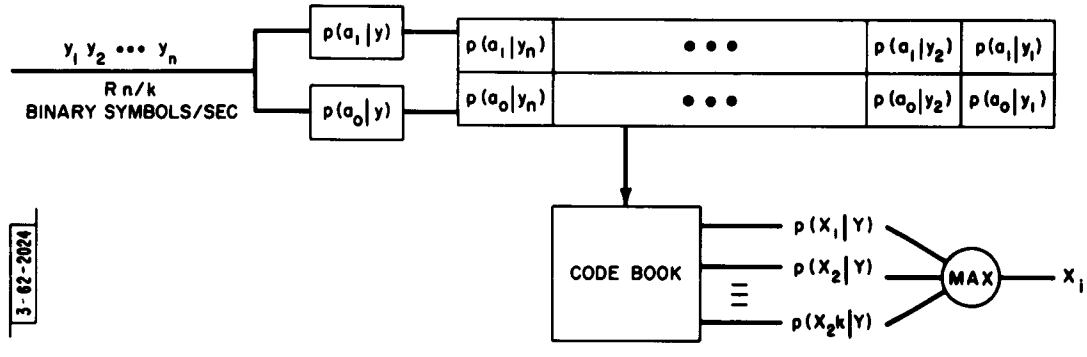


Fig. 9. Decoder for binary input-continuous output channel.

In the continuous channel, (Fig. 6) the receiver computed $P(X_i \mid Y)$ for each of the waveforms X_i. In the discrete input case each X_i is now a sequence of n elementary channel waveforms. Under the assumption that the channel is memoryless, $P(X_i \mid Y)$ can be expressed as the product of the corresponding probabilities of the symbols making up the sequence, i.e., $P(X_i \mid Y)$ is of the form $p(x_1 \mid y_1)\ p(x_2 \mid y_2) \ldots\ p(x_n \mid y_n)$, where the transmitted message is the sequence $x_i, x_2 \ldots x_n$ and the received message is the sequence $y_1, y_2, \ldots, y_n$. Each of the elementary signals x_i has one of two (in general 2^{ℓ}) forms. However each of the y_i is still obtained by adding a noise sample to the transmitted waveform. Thus y_i is still an arbitrary waveform of length τ.

The receiver (Fig. 9) makes its decision on a block basis. It receives n waveforms $y_1, y_2, \ldots, y_n$. The channel makes a tentative decision on each received symbol. A decoder which follows the channel makes a final decision on the block of n channel decisions. More precisely, when the first symbol y_1 of the block is received, the channel computes $p(a_0 \mid y_1)$ and $p(a_1 \mid y_1)$, i.e., the probabilities that the first transmitted symbol was a_0 and a_1 respectively. These two probabilities are stored in the decoder. The same computation is made for each of the n symbols of the block. The decoder, of course, knows the code, i.e., which 2^k of the 2^n possible sequences were selected. It computes the probabilities of these sequences by taking the appropriate products of the symbol probabilities and gives as its final output the sequence with highest probability. Suppose, for example, that $k = 2$ and $n = 3$ or equivalently that 4 of the 8 possible sequences of 3-digit binary symbols may be transmitted. Let these 4 sequences be

$$X_1 = 0\ 0\ 0$$
$$X_2 = 0\ 0\ 1$$
$$X_3 = 0\ 1\ 0$$
$$X_4 = 1\ 0\ 0$$

If the received sequence $Y = y_1, y_2, y_3$, then the decoder computes

$$p(X_1|Y) = p(a_0|y_1)\ p(a_0|y_2)\ p(a_0|y_3)$$
$$p(X_2|Y) = p(a_0|y_1)\ p(a_0|y_2)\ p(a_1|y_3)$$
$$p(X_3|Y) = p(a_0|y_1)\ p(a_1|y_2)\ p(a_0|y_3)$$
$$p(X_4|Y) = p(a_1|y_1)\ p(a_0|y_2)\ p(a_0|y_3)$$

and decodes the sequence with highest probability.

Another way of describing the same computation is in terms of a so-called "distance" or "metric". Clearly it makes no difference whether the receiver computes $p(X_i|Y)$ or some monotone function of $p(X_i|Y)$ say $D(X_i, Y) = -\log p(X_i|Y)$. Maximizing $p(X_i|Y)$ is equivalent to minimizing the distance $D(X_i, Y)$ [if $p(X_i|Y) = 1$, then $D(X_i, Y) = 0$]. Since $p(X_i|Y) = p(x_1|y_1)\ p(x_2|y_2), \ldots p(x_n|y_n)$, it follows that

$$D(X_i,Y) = -\log p(X_i|Y) = -\log p(x_1|y_1) - \log p(x_2|y_2), \ldots - \log p(x_n|y_n)$$
$$= d(x_1, y_1) + d(x_2, y_2) + \ldots + d(x_n, y_n).$$

Thus the "distance" between two sequences X and Y is the sum of the "distances" between the corresponding symbols making up the sequences.

Probability of Error

What can be said about the behavior of the probability of error for this channel with discrete input alphabet? The essential difference between this channel and the continuous channel is the fact that 2^k waveforms are selected out of 2^n possible waveforms of length T rather than out of the set of all functions of length T. One certainly cannot expect an improvement in performance by restricting the class of possible waveforms. It turns out that the general form of Shannon's theorem still holds, that is

$$P_e \sim 2^{-TE_2(R)}$$

where the new exponent E_2 is everywhere smaller than the exponent E_1 of the continuous case. The more the set of waveforms is restricted, the smaller the value of the exponent for each value of R. This is shown in Fig. 7.

Output Quantization

In the channel with discrete input the decoder stores 2n probabilities or "distances" obtained from the channel output and makes 2^k computations involving these 2n numbers. Not only is 2^k a frightening number

of computations for useful values of k, but in addition storage for $2n$ distances or probabilities may also be unattractive. One way of reducing the storage requirements is by "quantizing" the channel output probabilities. The channel with which we have been dealing in this section has a binary input alphabet but an output alphabet consisting of 2 continuously valuable probabilities or distances. If we use, say, 10 bits to specify each probability or distance (.1% accuracy) then 20 bits of storage are required to store a single channel output. If this represents too much storage for the decoder, then the distances may be stored with fewer bits and therefore with less precision. The binary symmetric channel represents an extreme case. Here we retain just one bit at the output which tells whether $p(a_0|y)$ is greater or smaller than $p(a_1|y)$. If $p(a_0|y) < p(a_1|y)$ the distance $d(a_0, y) = 0$ and $d(a_1, y) = 1$. In this case of maximum quantization the channel is said to make a "hard decision"; it calls the received symbol a_0 or a_1 according to which decision is more probable. The distance defined in this case is the so-called Hamming distance.

It seems reasonable that the less information the decoder retains on which to base its final decision, the poorer will be its performance. Again, even with channel output quantization, Shannon's theorem still takes the same form

$$P_e \sim 2^{-TE_3(R)}$$

where E_3 (shown in Fig. 7) is smaller than E_2, how much smaller depending upon how much information is discarded by the channel.

Central Problems of Coding Theory

What we have attempted to show to this point is that given a communications medium and a continuous channel using that medium, it is possible to use the channel reliably and efficiently (Shannon's theorem for continuous channels). To obtain the kind of performance that we want, continuous channels are impractical. We convert the continuous channels to discrete channels by input quantization (limiting the number of possible input symbols) and by output quantization (limiting the amount of information retained by the decoder per symbol). Both forms of quantization simplify the encoding and decoding equipments at the cost of performance.

One of the central problems in communication theory is finding discrete channels and codes which are both practical and which do not degrade the

performance excessively from the continuous channel. By finding channels and codes we mean finding input and output quantizations and then making "good" selections of sequences of channel symbols so that the resulting error exponent $E_3(R)$ does not depart excessively from the optimum exponent $E_1(R)$.

The other problem and the one to which we devote the remainder of this note, is the problem of decoding. Regardless of the channel quantization, the method of decoding described up until now has required 2^k comparisons at the decoder leading to the selection of the most probable sequence. As long as this "maximum likelihood" method of decoding is retained, this number of computations is fundamental. Another method of decoding has been found by Wozencraft which yields the same error exponent as maximum likelihood decoding but far less computation. This technique is called sequential decoding and its properties are described in the rest of this note.

Sequential or Convolutional Encoding

The essence of sequential decoding is the replacement of the "jumping" constraint of block coding by a "sliding" constraint. In an (n, k) block code, a block of k information bits is mapped into a block of n binary symbols by the generation of $n-k$ parity check symbols which depend upon the k information bits, (Fig. 8). The next block in sequence depends upon the next k information bits and thus is completely independent of all previous blocks. In a sequential code with the same parameters, check symbols are interspersed between successive information digits with each check symbol dependent upon the previous k information bits.

We show this in Fig. 10 for $k = 4$, $n = 8$, or $r = 1/2$. On line (a) we show an infinite stream of information bits m_i as emitted by an information source. On lines (b) and (c) we show the dependencies of the transmitted symbols by brackets. Line (b) represents a block coding scheme: message bits $m_4 - m_7$ are transmitted together with check digits $c_4 - c_7$ to form an 8 digit block. The four parity check digits depend only on $m_4 - m_7$. The four check digits $c_8 - c_{11}$ in the next block depend only on the message digits $m_8 - m_{11}$ of that block. (Clearly, with the dependency specified in this way it makes no difference whether the check digits are uniformly distributed in the block, clumped at one end, or arranged in any other way). On line (c) the sequential configuration is shown. Check digit c_4 is dependent upon $m_1 - m_4$, c_5 on $m_2 - m_5$, c_6 on $m_3 - m_6$, etc.

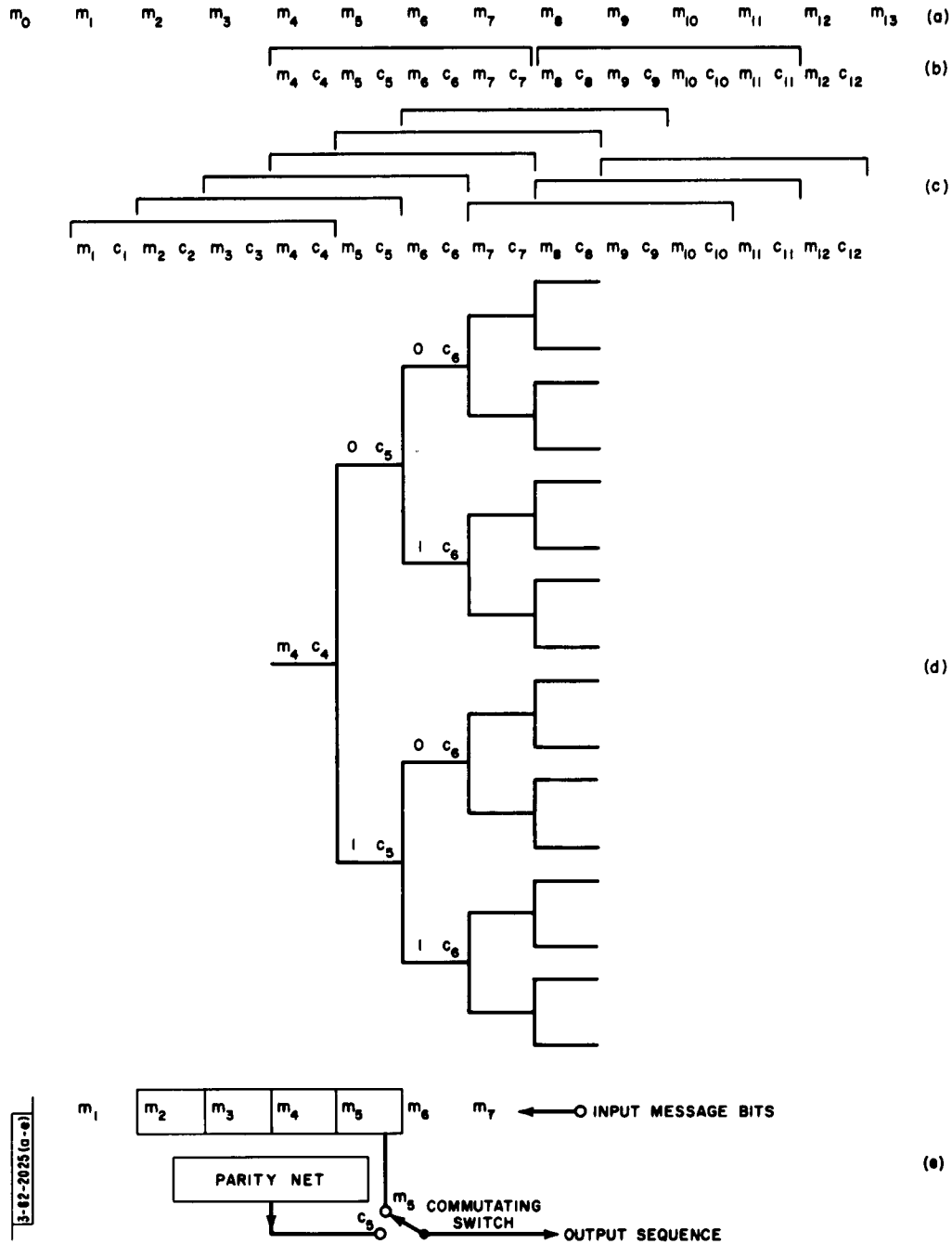


Fig. 10. Convolutional encoding.

It is convenient to think of the sequential encoding process in terms of a coding "tree". To see how this comes about we refer to Fig. 10d and begin by assuming that we have just taken m_4 from the source and generated check digit c_4. We have two choices for m_5 , 0 or 1. Once m_5 has been selected, the check digit c_5 is completely determined by m_5 and the previous message bits m_2 , m_3 and m_4 . This is represented in Fig. 10d as follows. Following m_4, c_4 is a branching corresponding to the two possible choices for m_5. The check symbol c_5 follows m_5 on each of the two branches. The arrival of m_6 causes another branching with the indicated dependence of c_6 on the four message bits m_3, m_4, m_5 and m_6. This process continues indefinitely.

Thus the sequential encoding process transforms an infinite sequence of information bits into an infinitely long tree with each branching corresponding to the two possible values of each information bit. The check digits appearing next to the information bit defining a branch are functions of that information bit and the $k-1$ preceding information bits. Figure 10d shows a tree for $r = 1/2$. If $r = 1/3$ then each message bit is followed by <u>two</u> check digits on each branch with each check digit a <u>different</u> function of the <u>same</u> k variables. In all cases, a <u>particular</u> input information sequence is transformed into a particular path through the tree.

A sequential code of this type is easily generated with a circuit similar to that of Fig. 10e. Message bits are shifted sequentially through a $k = 4$ storage cell register from right to left. A parity network connected to the register generates the check digits. In Fig. 10e, digits $m_2 - m_5$ are contained in the register, hence the parity network generates c_5. The output sequence is formed by switching alternately to the latest message digit in the rightmost cell of the register and to the parity output, the switching occurring at twice the input data rate, ($r = 1/2$). Since the output sequence is obtained by shifting a message sequence past a fixed parity network, the encoder is referred to as a <u>convolutional</u> encoder.

The process is begun by assuming an initial sequence of $k-1$, 0's preceding the message sequence. Thus the first check symbol c_1 is $c_1(0, 0, 0, m_1)$. In terms of the circuit of Fig. 11 this is accomplished by clearing the shift register before shifting in the first message digit.

All the above remarks and those which follow while stated for binary codes apply equally well to other alphabets and tree structures.

Some Properties of Trees

It is clear from the previous section that once parameters k and n (or k and r) have been established and the parity generating functions have been selected, the code tree is completely determined. Having fixed k and r, the problem of finding "good" codes is the problem of finding "good" parity networks. The "goodness" of a tree, or of the parity networks generating the tree, is determined by how "unlike" the paths through the tree are with respect to a decoding "distance", which as shown above is related to the channel conditional probabilities.

The tree repeats every k branches. This is seen in Fig. 10d. At m_9, the parity symbol c_9 depends on m_6, m_7, m_8 and m_9. It is therefore independent of the branching at m_5. Hence, if we compare two infinite message sequences s_1 and s_2 which are identical except for one bit, say m_5, the transmitted sequences (or paths) x_1 and x_2 corresponding to s_1 and s_2 are identical except for the branches from m_5 through m_8. If the two sequences s_1 and s_2 differ in two bits m_5 and m_8, the branches of x_1 and x_2 do not become identical until m_{12}. In general if two message sequences differ in some digit, the corresponding paths do not agree for at least $k - 1$ additional branches.

If s_1 and s_2 differ only at m_5, then the distance between x_1 and x_2 increases until m_9 is reached. Thereupon the distance remains constant.

Sequential Decoding - General Principles

As we have pointed out above, the sequential encoding process differs from block encoding in one major respect: the jumping constraint of block encoding is replaced by the sliding constraint of sequential or convolutional encoding. The result of the sequential encoding process is the transformation of an infinite information sequence into an infinitely long path through a tree. The sequential decoding process is therefore the reconstruction of the path corresponding to the transmitted sequence based upon the information contained in the received sequence. The criterion by which a sequential decoder selects the correct path differs from the criterion by which the block decoder selects the correct sequence in a fundamental way. The block decoder uses the criterion of maximum likelihood or some variation thereof. Sequential decoding imposes a threshold based upon an assumption about the noise characteristics. Thus, whereas block decoding looks for the most probable sequence, sequential

decoding looks for a sequence which is "sufficiently probable" with respect to some threshold.

It is these two fundamental differences between sequential and block operation that provide the basis for decoding or tree search algorithms which drastically reduce the number of decoding operations below the block code requirements while retaining the same error probability behavior as in the block code. The undesirable feature that results from these same properties is the catastrophic nature of decoding errors. That is, once an incorrect branch is chosen (without the opportunity to change the decision) the probability of the decoder correcting itself (finding the right path) is low.

Given the fundamental properties of sequential operation, i. e. coding trees and threshold decoding, how does a decoder operate? The basic physical law which governs this operation is the probabilistic law of large numbers which says grossly that the more observations made of a random process the more reliable are the inferences that can be made about the process. Every received channel symbol provides a sample of the channel noise. The longer the sequence of symbols that the decoder observes the more samples of channel noise are being observed and the more "typical" the noise should look. A short sequence of symbols may exhibit atypical noise; the longer the sequence the less probable it is that the noise looks atypical. Suppose that the decoder is on the correct path. An atypical noise event can make the correct path look "too improbable" for a short time. The longer the observation of the correct path the more likely it is to exhibit predicted behavior. On the other hand, when the decoder takes an incorrect path, in the absence of channel noise, the longer the observation the less probable the path appears. Again an atypical noise event can make an incorrect path appear more probable than the correct path. However, the longer the observed sequence the less probable atypical noise behavior becomes and the smaller the probability that an incorrect path looks "sufficiently probable". It also follows that the greater the distance that a decoder proceeds along an incorrect path the more computations will be required to rectify the error. Thus the probability of remaining on an incorrect path for a given path length decreases with the length and the number of operations to rectify the error increases with length. It turns out that one may find tree search schemes in which the average number of computations is low or equivalently schemes in which the probability of long incorrect paths is sufficiently small.

Sequential Decoding Algorithms

As the decoder proceeds to reconstruct a path through a tree based upon a received sequence, it compares the probability of the path that it is currently exploring against a threshold determined by the current noise level that the decoder expects. If the comparison is favorable (the path looks "sufficiently probable"), the decoder continues forward in the tree. If the comparison is unfavorable then either the decoder is on the wrong path because of an atypical noise event that occurred sometime previously or the decoder is on the correct path and an atypical noise event is now occurring.

At this point the decoder makes the assumption that the path is incorrect. The decoder reverses itself and searches back in the attempt to find a more probable path. The distance it is allowed to go back in this search depends upon the particular algorithm. For the moment assume that the decoder searches back until it either finds a "sufficiently probable" path or it retreats some fixed number d_0 nodes back without finding a good path. The former case implies that the first path was indeed incorrect as hypothesized and that the new path is more likely correct. The decoder thereupon proceeds forward. The latter case implies that the first path may still be correct and only appears improbable because of atypical noise behavior. The decoder relaxes its criterion of "sufficiently probable" and proceeds as before with a new threshold based upon the current assumption of the noise level.

The Wozencraft or SECO Algorithm

The remarks of the previous section are quite general and apply to any sequential decoding algorithm. We must now become more specific and pin down some of the parameters that define a decoding algorithm. In this section we consider the SECO algorithm which is a variation of the original Wozencraft scheme. In the next section we shall discuss the Fano decoding algorithm.

Three properties of a decoding algorithm which were left either vague or unmentioned in the previous section must be specified: first, the definition of "sufficiently probable" as applied to the decoding threshold; second, the number of nodes the decoder is allowed to search back in the attempt to find a good path before relaxing the threshold; and third, the criteria used to make final or irrevocable decisions about branching nodes, and the criteria for releasing the decoded digits to the customer.

In the SECO algorithm all three of these properties are tied to the constraint length k. Referring again to the tree of Fig. 10 d, suppose that the path through m_4 is correct and that the decoder is about to make a decision on m_5. We divide the tree into two subsets of equal size stemming from the two branches at m_5. We call the subset containing the correct path the correct subset and we call the subset stemming from the incorrect branch at m_5 the incorrect subset. The basic operation of the SECO procedure is to distinguish between the two subsets and thereby decode m_5. To do this we fix a small fraction, P_1 and then define a distance function $K_1(\ell)$ (Fig. 11) as follows: the probability that any sequence ℓ branches long in the incorrect subset will be closer to the corresponding subsequence of the received message than $K_1(\ell)$ is less than P_1. For a larger probability P_2 we obtain a function $K_2(\ell)$ which is larger than $K_1(\ell)$ for all values of ℓ.

The functions $K_j(\ell)$, $0 < \ell < k$, are then used as the criteria of the previous section. If a path of length ℓ is found closer to the received sequence than $K_1(\ell)$, this path is defined to be "sufficiently good" since the probability that this path is in the incorrect subset is less than P_1. When a path of length k is found to be "sufficiently good" then the message bit defining the beginning of that path is decoded. If a path of length k is not good enough, then the decoder goes backward and then forward taking alternate branches in a systematic search for a sequence of length k that meets the criterion. If the decoder has examined all branches back to that of the digit being decoded and has been unsuccessful, then the assumption is made that the noise is atypical, the decoder returns to its old path, relaxes the criterion from K_1 to K_2 and begins again. In subsequent decodings the criterion is lowered whenever possible. The criteria for releasing the decoded digits to the user are discussed later.

The Fano Decoding Algorithm

In the SECO algorithm, the final decoding of a digit and the tree search are intimately related. A digit is decoded whenever a good path k branches long stemming from that digit is found since that path belongs to the correct subset with very high probability. Fano's algorithm divorces the decoding from the tree search. His decoder always attempts to find the most probable path through the tree. When a digit is decoded it is determined by other conditions discussed later.

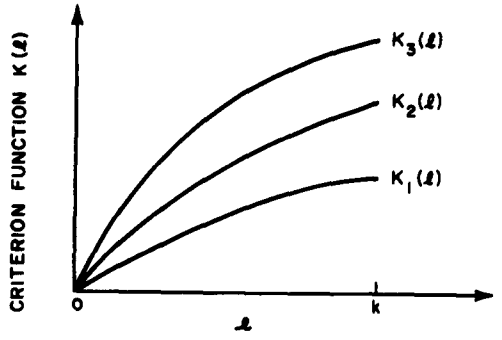


Fig. 11. SECO criterion functions.

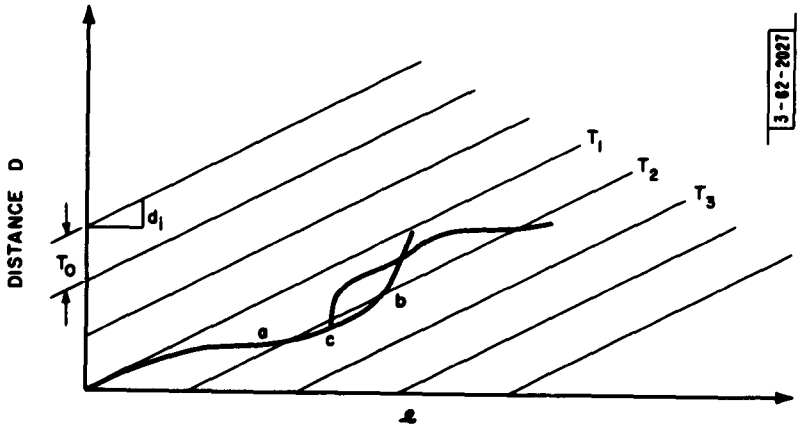


Fig. 12. Fano decoder criterion functions.

The criterion functions used in Fano decoding are shown in Fig. 12. These consist of a number of parallel lines of slope d_o and with adjacent intercepts differing by T_o . These curves are, of course, qualitatively similar to those of Fig. 11. Unlike SECO the abscissa ℓ designating path length is not limited by the constraint length but extends indefinitely. Plotted on the same curve is the distance of the current path from the received sequence. The current threshold is by definition the one immediately above the distance curve. From the origin to point a the decoder proceeds forward taking the better branch at each node and the threshold is T_1 . At point a the distance is increasing slowly enough (the noise level is low enough) for the distance to cross curve T_2 ; it continues past point c taking the lower (more probable) path until the distance starts increasing rapidly and at point b crosses the threshold curve. At this point the decoder assumes that the path is not sufficiently probable and it searches back changing decisions at branches in an attempt to find some path which remains below threshold T_2 . In this process it goes back as far as point a and failing to find an exit below curve T_2 , it returns to point b relaxing the criterion to threshold T_1 . The original path again fails, and the search this time yields the upper path at point c which proceeds below curve T_1 .

This procedure attempts to make maximum use of the law of large numbers, i. e., the fact that the longer the observed path the more probable the correct path will appear and the less probable all incorrect paths will appear. The parameters d_o and T_o are adjusted to give optimum performance of the decoder. Unlike the SECO algorithm, constraint length does not appear in in the tree search procedure nor is the final decoding related to the tree search.

Waiting Lines and Buffer Storage

There is no attempt in this note to derive any quantitative properties of sequential decoding algorithms. It should be clear, however, because of the basic structure of the sequential decoding process that if the average noise level is sufficiently small, then large noise perturbations will be sufficiently rare so that the decoder is on the correct path most of the time. The greater the noise level, the more common are the large perturbations and the more likely it is for the decoder to make false starts on incorrect paths and require searches to find the correct path. Wozencraft has shown that for rates below some maximum called R_{comp} , the average number of branches observed by the

decoder remains finite (For the binary symmetric channel, $1/2\ C \leq R_{comp} \leq C$ where C is the channel capacity.)

However, even with a small average number of computations, there are individual times during which the peak computational load may become large. It seems reasonable, therefore, to design a decoding machine which can handle the average computational load with a buffer storage to smooth out the computational peaks. Suppose that the decoder contains storage sufficient for N branches. We can represent this as in Fig. 13 by superimposing a window N branches long on a decoding tree. The storage represented by the window is used for a variety of purposes related to the execution of the decoding algorithm. In Fig. 13a, a number of nodes in the tree have been designated. Nodes P_1 and P_6 are respectively the oldest and newest nodes in the window. P_5 is the position of the latest branch to enter the decoder from the channel; P_3 is the current node under examination. P_4 is the farthest point to the right to which P_3 has advanced and P_2 is the farthest point to the left to which P_3 can go or, in other words, the oldest node which can still be examined. The distance between P_5 and either P_3 or P_4 is indicative of the waiting line, while the distance between P_2 and P_1 is some fixed delay between an irrevocable decision and release of data. The entire window (hence points P_1, P_2 and P_6) moves to the right at the rate at which digits are decoded or passed on to the user. P_5 moves to the right at the rate at which digits are received from the channel.

In Fig. 13b, the SECO configuration is shown. The distance between P_2 and P_4 is fixed at the constraint length k . The distance between P_3 and P_2 is the length ℓ of the path being examined (the abscissa of Fig. 12). The waiting line is defined to be the distance between P_5 and P_4 . A digit is decoded when the k-branch path between P_2 and P_4 looks "sufficiently probable". Thus the entire window moves to the right at the decoding rate while P_5 moves to the right at the uniform channel rate. When P_5 reaches P_6 the buffer overflows and the decoder must stop.

The Fano decoder is represented in Fig. 13c. Here we may consider P_5 and P_6 to coincide. Thus the decoding rate and the channel rate are identical and quite independent of the decoder or tree search operation. P_3 moves at the tree-search rate. Hence the buffer overflows when P_3 reaches its left-most point P_2 . The distance between P_1 and P_3 is the length ℓ of Fig. 4 and the delay d between P_1 and P_2 may be interpreted as the minimum allowable value of ℓ at which a decoding may occur.

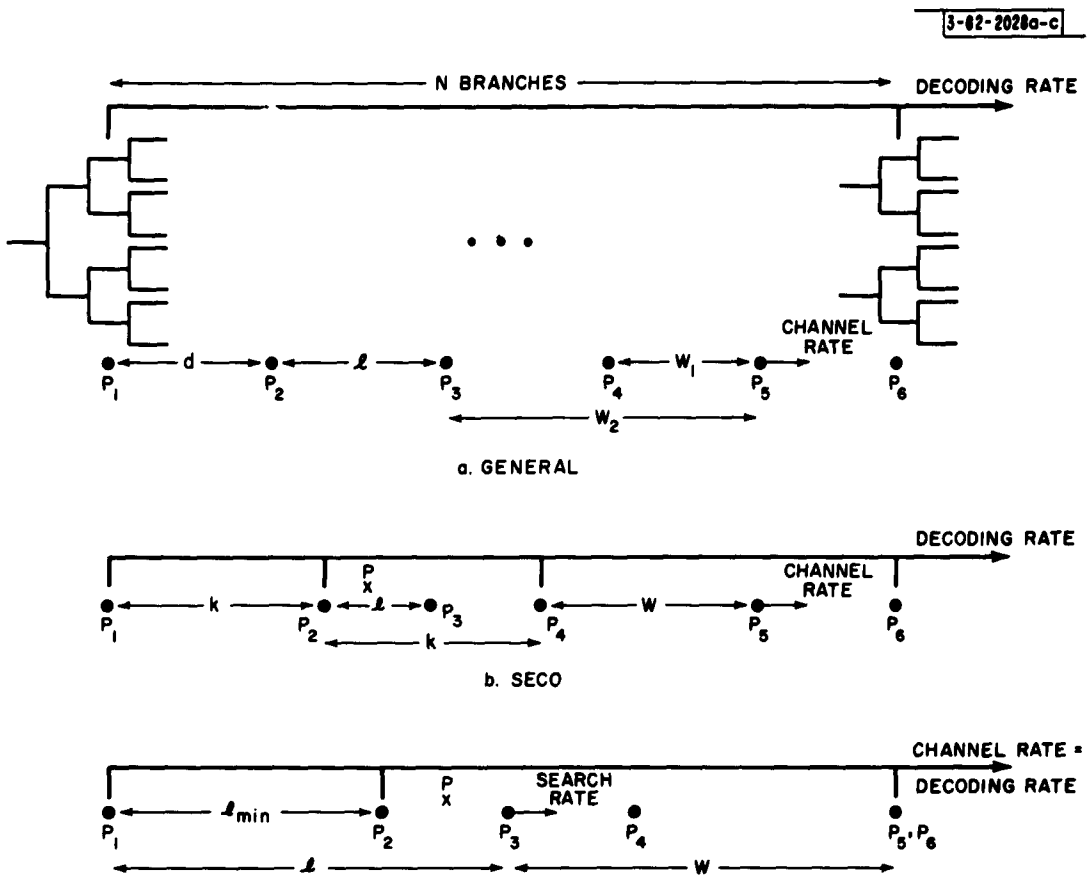


Fig. 13. Decoder storage.

In either algorithm, once P_2 moves past a node, the branch chosen at that node can no longer be changed. The delay d between P_2 and the final release at P_1 is provided for error detection. If the probability of an incorrect branching at P_2 is not sufficiently low, the delay provides additional time for detecting a possible error even though that error may no longer be corrected. In the SECO machine the delay is fixed at k, the constraint length. This restriction is not fundamental to the Wozencraft algorithm.

Probability of Error

According to the previous section, a digit is released when the branch corresponding to that decision leaves the decoder, or equivalently when the N branch window defining the decoder storage passes the branch. Thus a decoding error is made if an incorrect branch is passed by the window.

During the process of searching the tree, a noise event often causes an incorrect branch to look "sufficiently probable". A succession of noise events can make a succession of incorrect branches or an incorrect path look "sufficiently probable". As the decoder proceeds along an incorrect path one of two results must occur: either the decoder remains on an incorrect path in which case it must eventually look improbable (law of large numbers), or the decoder somehow stumbles back on the correct path and effectively returns to normal. The meaning of the latter is best explained with reference to the tree of Fig. 10. The tree structure repeats every k branches. Thus, for example, the tree beginning with the branch corresponding to message bit m_{10} is independent of the choice of m_6 and earlier message bits since $k = 4$. Consequently it is possible for a noise sequence to force an incorrect branching at m_6 leading to the same branching at m_{10} that would have been chosen had the correct branch at m_6 been taken.

There are, therefore, two types of decoding errors: type (1), an incorrect branching that continues to look probable and results in the decoder correcting itself and type (2), an incorrect branching which looks correct for a long enough time so that the error is passed on before it can be corrected.

With these remarks in mind, let us consider decoding errors in the SECO algorithm with reference to Fig. 13b. Suppose that the path up to node P is correct and that an incorrect branch is taken at P. This incorrect branch cannot be corrected (reaches P_2) if the subsequent noise is such that P_3 or P_4 advances k branches beyond P and the path looks "sufficiently

probable." The two error mechanisms are indistinguishable here since only k successful branches are required for a decoding.

Once P_2 has passed an incorrect branch at P , the best that can be achieved is detection of the error. A type 1 error is undetectable; a type 2 error may be detected before being passed by P_1 by virtue of the law of large numbers. One manifestation of this is an increase in the amount of searching or equivalently a decrease in the average rate at which the window moves to the right. If this is sufficiently slow P_5 will reach P_6 , or the buffer will overflow, before P is passed by P_1.

Another mechanism for error detection involves placing a confidence measure on the decoding criteria. In the SECO procedure each criterion curve K_j has associated with it a number P_j which is indicative of the probability of an incorrect branch being passed by P_2. The basic decoding procedure described above permits the decoder to use successively higher criterion curves (higher P_j) as the noise level increases, resulting in higher decoding error probabilities. An obvious detection procedure might then be to establish a maximum curve. A still more effective procedure is indicated in Fig. 14. Here we plot the decoding distance (over a constraint length) as a function of node number. In curve (a) the first (oldest) branch under consideration is correct; in curve (b) it is incorrect. The important fact is that it is possible to find a threshold below which the distance remains most of the time when on the correct path and above which the distance remains most of the time when on an incorrect path. Use of this threshold is preferable to fixing a maximum criterion curve since it allows excursions of the distance beyond the threshold provided these excursions are short enough. It thus allows the decoder to correct small error bursts that lead to high criterion curves but which can be subsequently handled by the decoder.

In the Fano decoder, the two types of errors are generally distinguishable. If the buffer is infinitely large so that the decoder can go back infinitely far to correct an incorrect decision, then type (2) errors will never occur since an incorrect path must eventually look improbable. Type (1) errors, of course, can never be detected. With a finite buffer, type (2) errors can occur. Suppose that in Fig. 13c an incorrect branch is taken at point P. The decoder (point P_3) proceeds to the right from P along an incorrect path. If the noise sequence is such that the P_2 passes point P before P_3 returns to P to correct the incorrect branching, then that error cannot be corrected. If P is later passed by P_1 before P_3 retreats to P_2 then that error is passed on

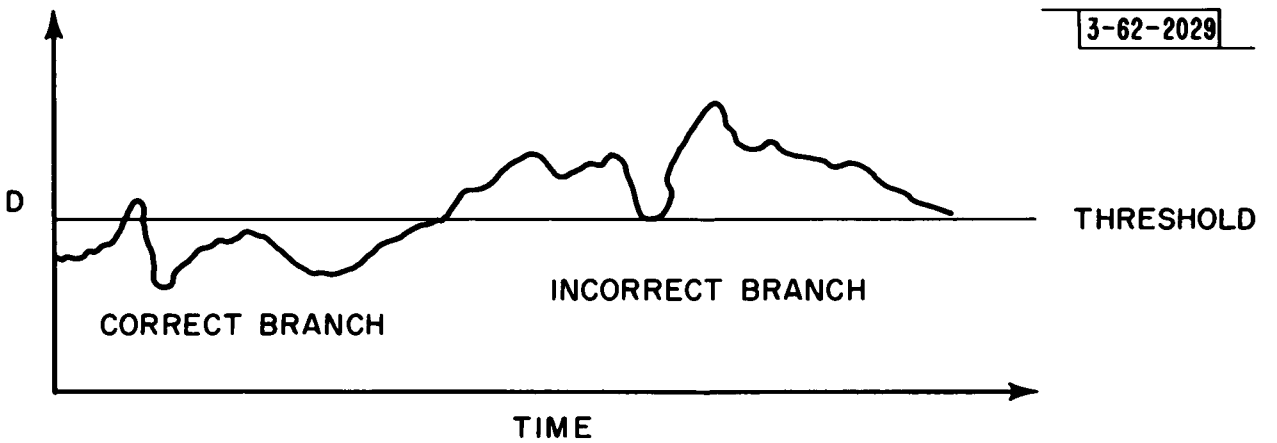


Fig. 14. SECO decoding distance-behavior with time.

to the user. Such errors may be detected with very high probability if the delay period between P_1 and P_2 is sufficiently large or by the application of a distance test similar to that of Fig. 14.

REFERENCES

1. J. M. Wozencraft and B. Reiffen, "Sequential Decoding," The Technology Press and John Wiley and Sons, Inc., New York (1961).

2. K. E. Perry and J. M. Wozencraft, "SECO: A Self Regulating Error Correcting Coder-Decoder," IRE Transactions on Information Theory IT-8, 5, pp. 128-135, (September 1962).

3. I. L. Lebow, P. G. McHugh, A. C. Parker, P. Rosen and J. M. Wozencraft, "Application of Sequential Decoding to High Rate Data Communication on a Telephone Line," IEEE Transactions on Information Theory, IT-9, 2, (April 1963).

4. R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding," IEEE Transactions on Information Theory, IT-9, 2, (April 1963).

5. G. Blustein and K. L. Jordan, Jr., "An Investigation of the Fano Sequential Decoding Algorithm by Computer Simulation," 62G-5 [U], Lincoln Laboratory, M.I.T., (July 1963).

DISTRIBUTION LIST

Director's Office

W. B. Davenport, Jr.

Group 28

L. Kleinrock
T. C. Bartee

Division 6

T. F. Rogers
G. P. Dinneen

Group 61

L. J. Ricardi

Group 62

R. S. Berg
G. Blustein
W. R. Crowther
N. L. Daggett
J. D. Drinan
J. A. Dumanian
H. E. Frachtman
J. E. Gillis
R. L. Givan
B. Gold
J. N. Harris
F. E. Heart
B. Howland
D. Huffman
B. H. Hutchinson
K. L. Jordan
R. S. Kennedy
I. L. Lebow (24)
L. D. Massey
A. A. Mathiasen
P. G. McHugh
N. J. Morrisson
F. Nagy
C. Niessen
A. C. Parker
F. G. Popp
C. M. Rader
P. Rosen
S. B. Russell
J. E. Savage
V. J. Sferrino
P. D. Smith
P. Stylos
R. Teoste
J. Tierney
R. V. Wood
Group 62 File (20)

Group 63

W. E. Morrow

Group 64

P. R. Drouilhet
T. J. Goblick
P. E. Green
J. L. Holsinger
M. J. Levin
J. G. Proakis

Group 65

L. I. Bluestein
B. J. Moriarty
B. Reiffen
H. Sherman
D. Wiggert
H. L. Yudkin
R. G. Enticknap

Group 66

J. R. Kinney
R. T. Prosser
E. Weiss

M. I. T., Cambridge

R. M. Fano
J. M. Wozencraft