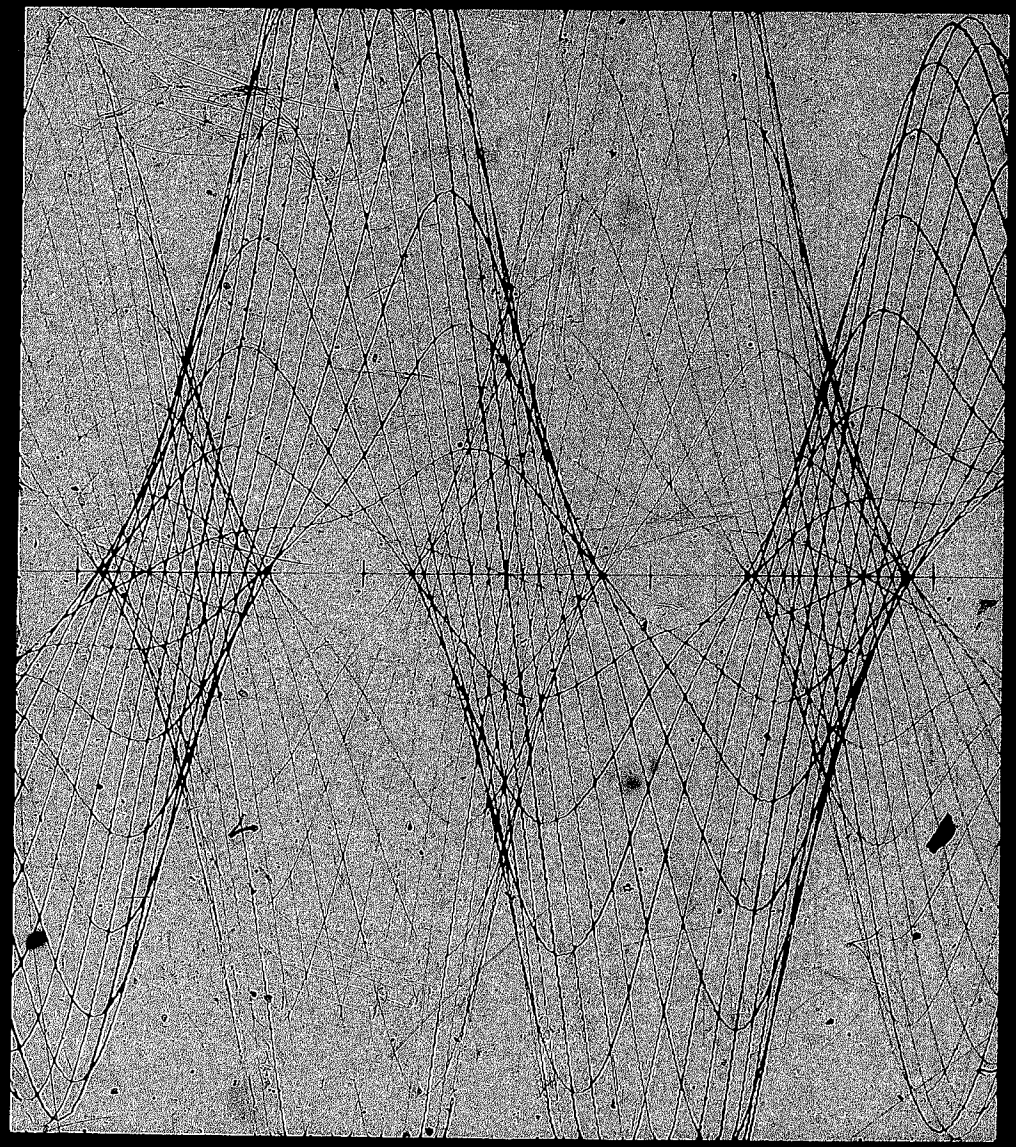

RC 2441

A STACK ALGORITHM FOR FASTER SEQUENTIAL DECODING OF TRANSMITTED INFORMATION

F. Jelinek

April 15, 1969

IBM RESEARCH

A STACK ALGORITHM FOR FASTER SEQUENTIAL DECODING OF TRANSMITTED INFORMATION.

F. Jelinek *
Research Division
Yorktown Heights, New York

ABSTRACT: In this paper a new Sequential Decoding algorithm is developed as a heuristically natural way of taking advantage of tree structure of codes. No previous knowledge of encoding is assumed. Our algorithm utilizes stack storage. It is much simpler to describe and analyze than the Fano algorithm, and is about six times faster than the latter at transmission rates equal to R_{comp}. Bounds on probability of error and on the average number of decoding steps are derived from scratch by previously known methods. Practical problems connected with instrumenting the stack algorithm are discussed and a scheme is described facilitating satisfactory performance even with limited stack storage capacity. Preliminary simulation results are presented estimating the decoding effort and the needed stack size.

RC-2441 ~~(# 11839)~~
April 15, 1969
Information technology (IR, Documentation, etc.)

* On sabbatical leave from the School of Electrical Engineering, Cornell University, Ithaca, New York.

1. INTRODUCTION

Sequential Decoding is a method of communication through noisy channels that utilizes tree codes [see Fig. 2 below]. Several decoding algorithms have been suggested, the two most important ones being due to Wozencraft[2] and to Fano[3] [see also Sec. 10.4 of Jelinek[1]]. The latter have the characteristic that when used with appropriate tree codes signalling over memoryless channels their probability of error decreases exponentially to zero at all transmission rates less than capacity C, while the average amount of decoding effort per decoded information bit is bounded above by a constant for all rates less than a quantity called R_{comp}. The latter is a function of the channel transmission probabilities only, and exceeds C/2 for all non-pathological memoryless channels. Figure 1 contains the plot of R_{comp}/C as a function of the crossover probability p of a binary symmetric channel. Sequential decoding can also be utilized as one component of a "hybrid" coding scheme[4,5,6] to achieve arbitrarily reliable communication at all rates less than C in exchange for a tolerable decoding effort. This is in contrast to all known practical methods of algebraic coding that achieve arbitrarily reliable performance only when the transmission rate is sufficiently reduced toward zero (in all fairness, it ought to be added that algebraic schemes are much simpler than sequential ones). Furthermore, algebraic methods work only for symmetrical channels with the same number of outputs as inputs (Forney[7] devised a scheme that gets around the last difficulty at the price of an increased decoder complexity).

In this paper we will introduce a new sequential algorithm that is faster than all competing ones, and that is very simple to describe and analyze. To realize this speed advantage without penalty in error probability, it is necessary to considerably increase the memory of the decoder. However, in a suitable environment (e.g. when a general purpose computer is used as a decoder) the increase in speed will be well worth the added cost.

2. TREE ENCODING

Let us assume that we desire to communicate over a discrete memoryless channel (d. m. c.) characterized by a transmission probability matrix $[w_o(\eta/\xi)]$, where $\xi \in \{0, 1, \ldots \alpha-1\}$ are the channel inputs and $\eta \in \{0, 1, \ldots, \beta-1\}$ are the corresponding channel outputs. Thus, for any integer n, the probability that an arbitrary output sequence $\eta_1, \eta_2, \ldots, \eta_n$ is received given that an arbitrary input sequence $\xi_1, \xi_2, \ldots, \xi_n$ was transmitted is equal to

$$\prod_{i=1}^{n} w_o(\eta_i/\xi_i)$$

Let us further assume that the information source generates outputs $s \in \{0, 1, \ldots, d-1\}$ that are to be communicated to the user located at the other end of the channel. An appropriate tree code of rate $R \triangleq \frac{1}{n_o} \log_2 d$ bits per channel use will have d branches leaving every node, each branch being associated with a sequence of n_o channel input digits. An example of a tree code appropriate for a binary source (d = 2) and a channel with binary inputs (α = 2) is given in Fig. 2. Since $n_o = 3$, the code has rate R = 1/3. The correspondence between source outputs and channel inputs (i. e. the encoding rule) is as follows:

At the start of the encoding process the encoder is "located" at the root of the tree. If the first digit s_1 generated by the source is a 0, the encoder follows the upper branch out of the root node to the next node and puts out (transmits through the channel) the sequence associated with that branch (in this case 000). If $s_1 = 1$, the encoder follows the lower branch to the next node and puts out the corresponding sequence (in this case 111). In general, just before the source generates the i th digit s_i, the encoder is located at some node i - 1 branches deep in the tree. If $s_i = 0$ the encoder leaves this node along the top branch, and if $s_i = 1$, it leaves the node along the bottom branch, and in both cases transmits the sequence corresponding to the branch along which it travelled. In this way a message sequence $s_1, s_2, \ldots$ traces a path through the tree and the tree sequence corresponding to that path is then transmitted. (The generalization to a d-nary tree is obvious: at time i the encoder leaves the current node along the $(s_i + 1)^{st}$ branch of the fan-out and transmits the corresponding sequence). Thus in Fig. 2, the message sequence 0011 determines the path indicated by the thick line and causes the sequence 0001011010 to be sent through the channel.

In principle, the encoding tree can be continued indefinitely and thus the total number Γ of levels it will have (the tree displayed in Fig. 2 has four levels) can be arbitrary. Since a d-nary tree with Γ levels has d^{Γ} paths, there is a problem of how the encoder can store its tree code. We will not concern ourselves here with that issue. Let the reader be assured that no difficulty arises. The tree is never actually stored, only an algorithm is that can generate the digits associated with the tree branches whenever the former are required. The usual tree codes are called convolutional and their description can be found on pp. 377 - 383 of reference 1. It will be simpler for us to continue to act as if the encoder stored the entire tree.

3. THE DECODING ALGORITHM

From the preceeding description of tree encoding it follows that the natural transmission units we are dealing with are not channel digits themselves, but sequences of n_o of these that correspond to the tree branches. Accordingly, it will simplify further discussion if we henceforth restrict our attention to the n_o-product channel $[w(y/x)]$ whose input symbols x and output symbols y are strings of n_o inputs and outputs of the underlying channel $[w_o(\eta/\xi)]$. From this point of view, the product channel input alphabet corresponding to the tree code of Fig. 2 is octal, and the message sequence 0011 causes 0532 to be transmitted. Let x^* represent some sequence $\xi_1^*, \xi_2^*, \ldots, \xi_{n_o}^*$ and let y^* represent $\eta_1^*, \eta_2^*, \ldots, \eta_{n_o}^*$. Then

$$w(y^*/x^*) \triangleq \prod_{i=1}^{n_o} w_o(\eta_i^*/\xi_i^*) \qquad (1)$$

Thus the product channel has inputs $x \in \{0, 1, \ldots, a-1\}$ and outputs $y \in \{0, 1, \ldots, b-1\}$ where $a = \alpha^{n_o}$, $b = \beta^{n_o}$, and α and β are the sizes of the input and output alphabets of the underlying channel, respectively. A tree path of length i is specified by the vector $\underset{\sim}{s}^i \triangleq (s_1, s_2, \ldots, s_i)$ (we will use bold face for vectors and superscripts will indicate their length) formed

from the corresponding message digits. We will denote by $x_j(\underset{\sim}{s}^i)$ $(j \le i)$ the transmitted symbol associated with the j^{th} branch of the path $\underset{\sim}{s}^i$. Thus, in Fig. 2, $x_3(0\ 1\ 0\ s_4) = 7$ for all s_4, and $x_2(1\ 0\ s_3\ s_4) = 2$ for all $s_3\ s_4$. Let us now assume that a sequence $\underset{\sim}{y}^\Gamma$ was received through the channel (Γ is the number of levels in the tree) and that we wish to <u>decode</u> this sequence, i.e. to determine the identity of the message sequence $\underset{\sim}{s}^\Gamma$ put out by the source. We will denote by $\underset{\sim}{\hat{s}}^\Gamma$ the receiver's estimate of $\underset{\sim}{s}^\Gamma$, and of course we aim at having $\underset{\sim}{\hat{s}}^\Gamma$ equal to $\underset{\sim}{s}^\Gamma$. We recall that when $\underset{\sim}{s}^\Gamma$ was inserted into the encoder, the latter produced the channel input sequence $\underset{\sim}{x}^\Gamma(\underset{\sim}{s}^\Gamma) = x_1(\underset{\sim}{s}^\Gamma),\ x_2(\underset{\sim}{s}^\Gamma), \ldots,$ $x_\Gamma(\underset{\sim}{s}^\Gamma)$ in the way described in the preceeding section. The block diagram of the communication process is given in Fig. 3, and our problem is to specify the operation of the <u>decoder</u>.

Let us make the the simplifying assumption that the source generates its successive outputs independently at random with a uniform probability. Then, if the decoder wishes to make its estimate $\underset{\sim}{\hat{s}}^\Gamma$ as reliable as possible, it ought to compute for each of the possible d^Γ messages the probability (random variables and vectors will be denoted by capitals).

$$\Pr\{\underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma \Big/ \underset{\sim}{x}^\Gamma(\underset{\sim}{s}^\Gamma)\} = \prod_{i=1}^{\Gamma} w(y_i \Big/ x_i(\underset{\sim}{s}^\Gamma)) \qquad (2)$$

that $\underset{\sim}{y}^\Gamma$ was received given that $\underset{\sim}{x}^\Gamma(\underset{\sim}{s}^\Gamma)$ was transmitted, and make its estimate $\underset{\sim}{\hat{s}}^\Gamma$ equal to that sequence $\underset{\sim}{s}^\Gamma$ which maximizes the probability (2). Since the search size d^Γ is enormous even for moderate values of Γ, this crudest of decoding methods is entirely out of the question. Rather, we would like to devise a scheme that would take advantage of the tree structure of the code. The idea would be to search the tree from its root out, trying to "fit" sequentially the beginning segments $\underset{\sim}{y}^i$ $(i=1, 2, \ldots, \Gamma)$ of the received sequence $\underset{\sim}{y}^\Gamma$ to the corresponding initial segments of the various paths of the tree.

Let $q(\underset{\sim}{s}^j) \triangleq \Pr\{\underset{\sim}{S}^j = \underset{\sim}{s}^j \Big/ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\}$ be the probability that the source generated the initial subsequence $\underset{\sim}{s}^j$ given that $\underset{\sim}{y}^\Gamma$ was received (for simplicity we suppress in the "q-notation" the dependence of the probability on the received sequence $\underset{\sim}{y}^\Gamma$ which remains invariant during the decoding of any given block). Then a reasonable tree search decoding algorithm would be as follows (we restrict ourselves initially to a binary tree, i.e. d=2):

1. Compute q(0) and q(1), the a posteriori probabilities that the decoder took the upper and the lower branch out of the root node (see Fig. 2).

2. If $q(0) \ge q(1)$, eliminate q(0) from the decoder's memory and compute q(00) and q(01). Otherwise, eliminate q(1) and

compute q(10) and q(11). Therefore, end up with the probabilities of three paths in the decoder's memory, two paths of length 2 and one of length 1.

3. Arrange the probabilities of the three paths in decreasing order. Take the path corresponding to the top most probability, compute the probabilities of its two possible one-branch extensions, and eliminate from memory the probability of the just extended path. [E. g. if q(0), q(10), q(11) are in the memory and, say, q(10) ≥ q(0) ≥ q(11), then q(10) is replaced in the memory by the newly computed values of q(100) and q(101). In case that, say, q(0) ≥ q(11) > q(10), then q(0) is replaced by q(00) and q(01).]. At the end of this step the decoder's memory will contain the probabilities of four paths, and either two of these will be of length 3 and one each of length 1 and 2, or all four paths will be of length 2.

4. The search pattern is now clear. After the k^{th} step, the decoder's memory will contain exactly k + 1 probabilities corresponding to paths of various lengths and <u>different</u> end nodes. The $(k+1)^{th}$ step will consist of finding the largest probability in the memory, determining the path to which it corresponds, and replacing that probability with the probabilities corresponding to the two one-branch extensions of that path.

5. The decoding process terminates when the path to be extended next is of length Γ, i. e. leads from the root node to the last level of the tree.

The above algorithm is a rather natural one since at any moment that path (from among those in the decoder's memory) will be extended which has the highest probability of being the true one. The feasibility of the algorithm will depend on two factors: (a) how difficult it is to compute the probabilities $q(\underset{\sim}{s}^i)$, and (b) how many computation steps are necessary for complete decoding (this number is proportional to the necessary size of the memory).

Let us determine the probability $q(\underset{\sim}{s}^j) = \Pr\{\underset{\sim}{S}^j = \underset{\sim}{s}^j \Big/ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\}$ that the source generated the initial subsequence $\underset{\sim}{s}^j$ given that $\underset{\sim}{y}^\Gamma$ was received. Using Bayes rule, the fact that the source outputs are uniformly distributed, and the relation (2), we get that

$$q(\underset{\sim}{s}^j) = \Pr\{\underset{\sim}{S}^j = \underset{\sim}{s}^j \Big/ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\} = \frac{\Pr\{\underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma \Big/ \underset{\sim}{S}^j = \underset{\sim}{s}^j\} \Pr\{\underset{\sim}{S}^j = \underset{\sim}{s}^j\}}{\Pr\{\underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\}} =$$

$$= \frac{\Pr\{\underset{\sim}{Y}^j = \underset{\sim}{y}^j \Big/ \underset{\sim}{x}^j(\underset{\sim}{s}^j)\} \Pr\{Y_i = y_i,\ i=j+1, \ldots, \Gamma \Big/ \underset{\sim}{S}^j = \underset{\sim}{s}^j\} d^{-j}}{\Pr\{\underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\}} =$$

$$= \left\{ \prod_{i=1}^{j} \left[w(y_i / x_i(\underset{\sim}{s}^j))\, d^{-1} \right] \right\} \left\{ \frac{\Pr\{ \underset{\sim}{Y}_i = \underset{\sim}{y}_i,\ i=j+1, \ldots, \Gamma / \underset{\sim}{S}^j = \underset{\sim}{s}^j \}}{\Pr\{ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma \}} \right\}$$

The first factor in braces on the righthand side of (3) is easily computable, but the second factor is a problem. We would like, therefore, to approximate the latter in some convenient manner. Let us recall the well known fact that in block coding, good codes can be selected at random with high probability (see p. 157 of Reference 1). We will see below that good tree codes can similarly be selected. To be precise, one would proceed as follows:

> Let $r(x)$, $x \in \{0, 1, \ldots, a-1\}$ be a probability distribution. Then select the symbols $x_i(\underset{\sim}{s}^\Gamma)$ associated with all of the $\sum_{j=1}^{\Gamma} d^j$ different branches of the tree at random, independently of each other, with the probability
>
> $$\Pr\{ x_i(\underset{\sim}{s}^\Gamma) = x \} = r(x) \qquad (4)$$

Suppose our tree code has been selected as above (What actual distribution $r(x)$ should be used will be discussed later. For the binary symmetric channel the correct distribution is the uniform one). Then, because of the law of large numbers, the probability

$$\Pr\{ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma \} = d^{-\Gamma} \sum_{\underset{\sim}{s}^\Gamma} \prod_{i=1}^{\Gamma} w(y_i / x_i(\underset{\sim}{s}^\Gamma)) \qquad (5)$$

will be very nearly equal to

$$\sum_{\underset{\sim}{x}^\Gamma} \prod_{i=1}^{\Gamma} [w(y_i / x_i)\, r(x_i)] = \prod_{i=1}^{\Gamma} \sum_{x=0}^{a-1} w(y_i / x)\, r(x) \qquad (6)$$

Similarly, since the symbols $y_{j+1}, \ldots, y_\Gamma$ depend on $\underset{\sim}{s}^j$ only because they can be caused by the sequences $x_{j+1}, \ldots x_\Gamma$ associated with the various paths stemming from the terminal node of the path $\underset{\sim}{s}^j$, we get the approximation

$$\Pr\{ Y_i = y_i,\ i=j+1, \ldots, \Gamma / \underset{\sim}{S}^j = \underset{\sim}{s}^j \} \approx \prod_{i=j+1}^{\Gamma} \left[\sum_{x=0}^{a-1} w(y_i / x)\, r(x) \right] \qquad (7)$$

As a consequence,

$$\frac{\Pr\{ Y_i = y_i,\ i=j+1, \ldots, \Gamma / \underset{\sim}{S}^j = \underset{\sim}{s}^j \}}{\Pr\{ \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma \}} \approx \frac{1}{\prod_{i=1}^{j} w_{\underset{\sim}{r}}(y_i)} \qquad (8)$$

where we have used the definition

$$w_{\underset{\sim}{r}}(y) \triangleq \sum_{x=0}^{a-1} w(y/x)\, r(x) \qquad y \in \{0, 1, \ldots, b-1\}$$

Substituting the approximation (8) into (3), taking logarithms of both sides, and using the definition of the transmission rate $n_o R = \log d$, we get

$$\log_2 q(\underset{\sim}{s}^j) = \log_2 Pr\{\underset{\sim}{S}^j = \underset{\sim}{s}^j / \underset{\sim}{Y}^\Gamma = \underset{\sim}{y}^\Gamma\} \approx$$

$$\approx \sum_{i=1}^{j} \left[\log_2 \frac{w(y_i/x_i(\underset{\sim}{s}^j))}{w_{\underset{\sim}{r}}(y_i)} - n_o R\right] \qquad (9)$$

We will use the approximation (9) to carry out the tree search algorithm described at the beginning of this section and take advantage of the fact that logarithms preserve ordering by size. Note that when extending the path $\underset{\sim}{s}^j$ to $\underset{\sim}{s}^{j+1} = (\underset{\sim}{s}^j, s_{j+1})$ one simply adds to the sum (9) the term

$$\lambda_{j+1}(\underset{\sim}{s}^{j+1}) \triangleq \left[\log \frac{w(y_{j+1}/x_{j+1}(\underset{\sim}{s}^{j+1}))}{w_{\underset{\sim}{r}}(y_{j+1})} - n_o R\right] \qquad (10)$$

corresponding to the last branch of the extended path $\underset{\sim}{s}^{j+1}$.

We shall now establish the usual nomenclature and then recapitulate the search algorithm. The sum

$$L(\underset{\sim}{s}^j) \triangleq \sum_{i=1}^{j} \left[\log \frac{w(y_i/x_i(\underset{\sim}{s}^j))}{w_{\underset{\sim}{r}}(y_i)} - n_o R\right] \qquad (11)$$

is called the likelihood function of the path $\underset{\sim}{s}^j$. The individual bracketed terms of the the sum, simply denoted by λ_i, (see (10)) are called branch likelihoods. Because of the ordered nature of its contents, the memory of the decoder is referred to as a stack.

1. At the beginning of the decoding process, the stack contains only the root node of the tree (i.e. the empty path) with its likelihood set arbitrarily at zero.

2. A decoding operation consists of ascertaining the path $\underset{\sim}{s}^j$ corresponding to the likelihood L_j at the top of the stack (i.e. the largest of the likelihoods in the stack), eliminating L_j from the stack, computing the likelihoods $\lambda_{j+1}^1, \ldots, \lambda_{j+1}^d$ of the branches that leave the end node of the path $\underset{\sim}{s}^j$, and reinserting the new path likelihoods $L_{j+1}^i = L_j + \lambda_{j+1}^i$, $i=1, 2, \ldots, d$, into their proper position according to size (clearly, the stack contains path identifications and their likelihood values).

3. The search ends when the decoder finds at the top of the stack a path whose length is Γ. That path is then considered to have been taken by the encoder.

In the next section, we will perform some analysis of our algorithm to estimate how large a stack is needed and what the probability of decoding error is. In Section 5 we will return to the algorithm, modify it so that the stack will grow by at most one entry after each decoding step (regardless of the number d of branches leaving a node), and examine the problem of stack maintenance. In the last section we will state some simulation results, make comparisons with Fano's algorithm, and specify a method of handling stack overflow.

We conclude this Section with an example. Consider the binary tree of Fig. 4 with nodes as numbered. Let paths be associated with their terminal nodes. Let the numbers written on top of the branches represent the values of the corresponding branch likelihood function for some received sequence y^3. Thus, the likelihood of path 8 is equal to -6 +1 +1 = -4. The state of the stack during decoding would then be as follows (top most path on the left).

1^{st} state : 0
2^{nd} state : 2, 1
3^{rd} state : 5, 6, 1
4^{th} state : 6, 1, 11, 12
5^{th} state : 1, 11, 12, 13, 14
6^{th} state : 3, 11, 12, 13, 14, 4
7^{th} state : 8, 11, 12, 13, 14, 7, 4

When trying to perform the eighth step, the decoder would find path 8 on top of the stack, and since the latter has length 3(=Γ), decoding would terminate.

4. PROBABILITY OF ERROR AND AVERAGE NUMBER OF DECODING STEPS

We now wish to compute upper bounds to the probability of decoding error and to the average number of decoding steps. There is nothing novel about our analysis[1, 3], and we present it for completeness only.

It will be convenient to partition the nodes of the tree into sets $\mathcal{D}_i$, $i = 0, 1, \ldots, \Gamma$ defined relative to the path $\underset{\sim}{s}^{\Gamma}$ actually taken by the encoder.

Definition 1:

The incorrect subset $\mathcal{D}_i$ consists of the end node of the initial segment $\underset{\sim}{s}^{i}$ of the true path $\underset{\sim}{s}^{\Gamma} = (\underset{\sim}{s}^{i}, s_{i+1}, \ldots, s^{\Gamma})$, of the d-1 terminal nodes of the incorrect branches $s^{*}_{i+1} \neq s_{i+1}$ stemming from the end node of $\underset{\sim}{s}^{i}$, and of all nodes lying on paths leaving these d-1 nodes.

A binary tree illustration of the sets $\mathcal{D}_i$ is given in Fig. 5. Let N_i be the number of nodes belonging to $\mathcal{D}_i$ that the decoder "visits" (these are end nodes of paths that have at one time appeared on top of the stack and were thus extended during the decoding process). Then since $\bigcup_{i=0}^{\Gamma} \mathcal{D}_i$ is the set of all nodes in the tree, $\sum_{i=0}^{\Gamma} N_i$ is equal to the total number of decoding steps and we will want to estimate it.

A decoding error will occur if there appears on top of the stack a path $\underset{\sim}{s}_{*}^{\Gamma}$ corresponding to any of the Γ - level (incorrect) nodes of any of the sets $\mathcal{D}_0, \ldots, \mathcal{D}_{\Gamma-1}$ before the true path $\underset{\sim}{s}^{\Gamma}$ appears. If $\underset{\sim}{s}_{*}^{\Gamma} \in \mathcal{D}_i$, we say that an error took place on level i + 1, since by Definition 1 the initial segment $\underset{\sim}{s}^{i}$ of $\underset{\sim}{s}^{\Gamma}$ is also the initial segment of $\underset{\sim}{s}_{*}^{\Gamma}$, while the $(i+1)^{th}$ digit s^{*}_{i+1} of $\underset{\sim}{s}_{*}^{\Gamma}$ differs from the corresponding digit of the true path. A necessary (but not a sufficient!) condition for an error on level i + 1 is that the likelihood $L(\underset{\sim}{s}_{*}^{\Gamma})$ (defined in (11)) be greater than or equal to the minimum value of the likelihoods $\{L(\underset{\sim}{s}^{i+1}), \ldots, L(\underset{\sim}{s}^{\Gamma})\}$ corresponding to the initial segments of the true path.

Thus the probability, $P_e(i)$, of (i + 1) - level error is bounded by

$$P_e(i) \leq \Pr\left\{ \bigcup_{\underset{\sim}{s}_{*}^{\Gamma} \in \mathcal{D}_i} \left[L(\underset{\sim}{s}_{*}^{\Gamma}) \geq \min_{i<m\leq\Gamma} L(\underset{\sim}{s}^{m}) \right] \right\} =$$

$$= \Pr\left\{ \bigcup_{\underset{\sim}{s}_{*}^{\Gamma} \in \mathcal{D}_i} \left[\bigcup_{m=i+1}^{\Gamma} \{ L(\underset{\sim}{s}_{*}^{\Gamma}) \geq L(\underset{\sim}{s}^{m}) \} \right] \right\} \leq$$

$$\leq \sum_{m=i+1}^{\Gamma} \Pr\left\{ \bigcup_{\underset{\sim}{s}_{*}^{\Gamma} \in \mathcal{D}_i} \left[L(\underset{\sim}{s}_{*}^{\Gamma}) \geq L(\underset{\sim}{s}^{m}) \right] \right\} \qquad (12)$$

and the probability of error P_e is bounded by

$$P_e \leq \sum_{i=0}^{\Gamma-1} P_e(i) \tag{13}$$

It is obvious that if the tree remains regular up to the last level (i.e. has always d branches leaving each node with one symbol x corresponding to each branch), then the probability $P_e(i)$ will be a monotonically increasing function of i, and in fact, $P_e(\Gamma-1)$ will be prohibitively high since it is simply equal to the probability that the branch likelihood of the last correct branch is less than the maximal likelihood of the (d-1) incorrect branches stemming from $\underset{\sim}{s}^{\Gamma-1}$. A simple expedient to assure that $P_e(i)$ remains acceptably small even for large i (close to $\Gamma-1$) is to associate with the last level branches not one but rather t+1 channel input symbols x where t is chosen suitably large (this is illustrated in Fig. 6 where t = 3). In our forthcoming analysis, we will assume that such an elongation took place. The net transmission rate will thus be reduced from R to $R\Gamma/(\Gamma+t)$ which is a negligible difference if $\Gamma >> t$. Before we bound the terms on the right-hand side of (12), let us return to N_i, the number of decoding steps in the incorrect subset $\mathcal{A}_i$. Obviously, a path $\underset{\sim}{s}^j_* \in \mathcal{A}_i$ $(i < j \leq \Gamma)$ will not appear on top of the stack unless its likelihood $L(\underset{\sim}{s}^j_*)$ is greater than or equal to the minimum of the true likelihoods $\{L(s^{i+1}), \ldots, L(s^{\Gamma})\}$. (Again, this condition is not sufficient for $\underset{\sim}{s}^j_*$ to be actually reached.) Let $\varphi(A)$ denote the indicator function of the event A, i.e.

$$\varphi(A) \triangleq \begin{cases} 1 \text{ if event A takes place} \\ 0 \text{ if event A does not take place} \end{cases} \tag{14}$$

Then

$$N_i \leq 1 + \sum_{j=i+1}^{\Gamma} \sum_{\underset{\sim}{s}^j_* \in \mathcal{A}_i} \varphi\left[L(\underset{\sim}{s}^j_*) \geq \min L(\underset{\sim}{s}^m)\right] \leq$$

$$\leq 1 + \sum_{j=i+1}^{\Gamma} \sum_{\underset{\sim}{s}^j_* \in \mathcal{A}_i} \sum_{m=i+1}^{\Gamma} \varphi\left[L(\underset{\sim}{s}^j_*) \geq L(\underset{\sim}{s}^m)\right] \tag{15}$$

where the number 1 accounts for the visit to the true node of $\mathcal{A}_i$. We are interested in the average number N_i of steps, and, therefore, in the quantities

$$\underset{\sim}{E}\left[\varphi[L(\underset{\sim}{s}^j_*) \geq L(\underset{\sim}{s}^m)]\right] \tag{16}$$

Now for any given code, the expectation in (16) involves averaging over the channel transmission process and over the message sequence generation process. As usual, it is not possible to estimate the quantity (16) conveniently for a fixed code, but it will be relatively easy to estimate the average of

(16) over the code ensemble generated by the random selection rule (4). We now proceed to do so.

It follows directly from (14) that for all $\sigma \geq 0$

$$\varphi\,[L(\underset{\sim}{s}^j_*) \geq L(\underset{\sim}{s}^m)] \leq \exp_2 \sigma[L(\underset{\sim}{s}^j_*) - L(\underset{\sim}{s}^m)] \qquad (17)$$

It turns out to be convenient to pick $\sigma = \frac{1}{2}$. Then, using definition (11),*

$$\underset{\sim}{E}\,[\varphi\,[L(\underset{\sim}{s}^j_*) \geq L(\underset{\sim}{s}^m)]] \leq$$

$$\leq 2^{\frac{1}{2}(m-j)n_o R} \underset{\sim}{E}\left[\frac{w(\underset{\sim}{Y}^j/\underset{\sim}{X}^j(\underset{\sim}{s}^j_*))}{w_{\underset{\sim}{r}}(\underset{\sim}{Y}^j)} \frac{w_{\underset{\sim}{r}}(\underset{\sim}{Y}^m)}{w(\underset{\sim}{Y}^m/\underset{\sim}{X}^m(\underset{\sim}{s}^m))}\right]^{\frac{1}{2}} \qquad (18)$$

where we are using the natural definitions

$$w(\underset{\sim}{y}^j/\underset{\sim}{x}^j) \triangleq \prod_{i=1}^{j} w(y_i/x_i) \qquad w_{\underset{\sim}{r}}(\underset{\sim}{y}^j) \triangleq \prod_{i=1}^{j} w_{\underset{\sim}{r}}(y_i) \qquad (19)$$

Let $n = \max\{m, j\}$ and suppose that $\underset{\sim}{s}^j_* \in \mathcal{B}_i$. Then the expectation on the righthand side of (18) can be rewritten as

$$\sum_{\underset{\sim}{y}^{n-i},\ \underset{\sim}{x}^{m-i},\ \underset{\sim}{x}^{j-i}_*} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{n-i}) \frac{w(\underset{\sim}{y}^{m-i}/\underset{\sim}{x}^{m-i})\, r(\underset{\sim}{x}^{m-i})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m-i})} r(\underset{\sim}{x}^{j-i}_*) \times$$

$$\times \left[\frac{w(\underset{\sim}{y}^{j-i}/\underset{\sim}{x}^{j-i}_*)}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{j-i})} \frac{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m-i})}{w(\underset{\sim}{y}^{m-i}/\underset{\sim}{x}^{m-i})}\right]^{\frac{1}{2}} =$$

$$= \sum_{\underset{\sim}{y}^{n-i}} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{n-i}) \left\{\sum_{\underset{\sim}{x}^{m-i}} \left[\frac{w(\underset{\sim}{y}^{m-i}/\underset{\sim}{x}^{m-i})}{w_r(\underset{\sim}{y}^{m-i})}\right]^{\frac{1}{2}} r(\underset{\sim}{x}^{m-i})\right\} \times$$

$$\times \left\{\sum_{\underset{\sim}{x}^{j-i}_*} \left[\frac{w(\underset{\sim}{y}^{j-i}/\underset{\sim}{x}^{j-i}_*)}{w(\underset{\sim}{y}^{j-i})}\right]^{\frac{1}{2}} r(\underset{\sim}{x}^{j-i})\right\} \qquad (20)$$

where $r(\underset{\sim}{x}^j) \triangleq \prod_{i=1}^{j} r(x_i)$,

(Note that $x_k(\underset{\sim}{s}^m) = x_k(\underset{\sim}{s}^j_*)$ for $k = 1, 2, \ldots i$ and that for $k > i$, $x_k(\underset{\sim}{s}^m)$ and $x_k(\underset{\sim}{s}^j_*)$ are picked independently of each other. Also,

*Because the last level tree branches have t+1 symbols, (18) is strictly true only for $m \neq n$. The necessary adjustment for the last level is trivial and we will omit it.

the received sequence $\underset{\sim}{y}^n$ is ...dependent of the symbols $x_k(\underset{\sim}{s}^j_*)$, $k > i$, that were not transmitted). Using Schwartz's inequality $\underset{\sim}{E}[|XY|] \le (\underset{\sim}{E}[X^2]\ \underset{\sim}{E}[Y^2])^{\frac{1}{2}}$ (20) can be bounded by

$$\left(\sum_{\underset{\sim}{y}^{n-i}} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{n-i}) \left\{\sum_{\underset{\sim}{x}^{m-i}} \left[\frac{w(\underset{\sim}{y}^{m-i}/\underset{\sim}{x}^{m-i})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m-i})}\right]^{\frac{1}{2}} r(\underset{\sim}{x}^{m-i})\right\}^2\right)^{\frac{1}{2}} \times$$

$$\times\left(\sum_{\underset{\sim}{y}^{n-i}} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{n-i}) \left\{\sum_{\underset{\sim}{x}^{j-i}_*} \left[\frac{w(\underset{\sim}{y}^{j-i}/\underset{\sim}{x}^{j-i}_*)}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{j-i})}\right]^{\frac{1}{2}} r(\underset{\sim}{x}_*^{j-i})\right\}^2\right)^{\frac{1}{2}} =$$

$$=\left(\sum_{y} w_{\underset{\sim}{r}}(y) \left\{\sum_{x}\left[\frac{w(y/x)}{w_{\underset{\sim}{r}}(y)}\right]^{\frac{1}{2}} r(x)\right\}^2\right)^{\frac{1}{2}(m-i)} \times$$

$$\times\left(\sum_{y} w_{\underset{\sim}{r}}(y) \left\{\sum_{x_*}\left[\frac{w(y/x_*)}{w_{\underset{\sim}{r}}(y)}\right]^{\frac{1}{2}} r(x_*)\right\}^2\right)^{\frac{1}{2}(j-i)} =$$

$$=\left(\sum_{y}\left\{\sum_{x} w(y/x)^{\frac{1}{2}} r(x)\right\}^2\right)^{\frac{1}{2}(m+j-2i)} \qquad (21)$$

It is now convenient to define the quantity

$$E_o(\sigma, \underset{\sim}{r}) \triangleq -\log_2\left(\sum_y \left\{\sum_x w(y/x)^{\frac{1}{1+\sigma}} r(x)\right\}^{1+\sigma}\right) \qquad (22)$$

so that the righthand side of (21) is $\exp_2\{-\frac{1}{2}(m+j-2i)\ E_o(1, \underset{\sim}{r})\}$. Substituting the bound (21) into (18) and the latter into (15), we get that

$$\underset{\sim}{E}[\bar{N}_i] \le 1 + \sum_{j=1}^{\Gamma-i}\sum_{m=1}^{\Gamma-i} 2^{j n_o R}\ 2^{\frac{1}{2}(m-j)n_o R}\ 2^{-\frac{1}{2}(m+j)E_o(1,\underset{\sim}{r})} =$$

$$= 1 + \left[\sum_{j=1}^{\Gamma-i} 2^{-\frac{1}{2}j(E_o(1,\underset{\sim}{r}) - n_o R)}\right]^2 \qquad (23)$$

where we have utilized the fact that the number of different nodes $\underset{\sim}{s}^j_* \in \mathcal{S}_i$ is equal to

$$(d-1)\ d^{j-i-1} \le 2^{(j-1)n_o R}$$

The bound (23) involves a sum of a geometric series that is going to be bounded by a constant independent of Γ and i provided

$$E_o(1, \underset{\sim}{r}) > n_o R \qquad (24)$$

Our next problem is to upper bound the probabilities in (12) averaged over the code ensemble (4). Since the probability

of an event is equal to the expectation of its indicator function (see (14)), we get that

$$\underset{\sim}{E}\left[\Pr\left\{\bigcup_{\underset{\sim}{s}^{\Gamma}_{*}\in\partial_i}\left[L(\underset{\sim}{s}^{\Gamma}_{*}) \geq L(\underset{\sim}{s}^{m})\right]\right\}\right] =$$

$$\underset{\sim}{E}\left[\varphi\left(\bigcup_{\underset{\sim}{s}^{\Gamma}_{*}\in\partial_i}\left[L(\underset{\sim}{s}^{\Gamma}_{*}) \geq L(\underset{\sim}{s}^{m})\right]\right)\right] \leq$$

$$\leq \underset{\sim}{E}\left[\left(\sum_{\underset{\sim}{s}^{\Gamma}_{*}\in\partial_i} \exp_2 \sigma\left[L(\underset{\sim}{s}^{\Gamma}_{*}) - L(\underset{\sim}{s}^{m})\right]\right)^{\nu}\right] \qquad (25)$$

for all $\sigma \geq 0$ and $\upsilon \geq 0$. The inequality in (25) is valid since all terms in the sum are non-negative and if for some $\underset{\sim}{s}^{\Gamma}_{*}$, $L(\underset{\sim}{s}^{\Gamma}_{*}) \geq L(\underset{\sim}{s}^{m})$ then the corresponding term exceeds 1, the maximum of an indicator function. It is convenient to set $\sigma = (1+\nu)^{-1}$. Then, using (11) and the reasoning that led from (18) to (30), the righthand side of (25) is upper bounded by (here we take into account the t + 1 symbols of the last level branches by defining $\Gamma^* \triangleq \Gamma + t$)

$$\sum_{\underset{\sim}{y}^{\Gamma},\,\underset{\sim}{x}^{m}} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{\Gamma^*})\,\frac{w(\underset{\sim}{y}^{m}/\underset{\sim}{x}^{m})\,r(\underset{\sim}{x}^{m})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m})} \times$$

$$\times \underset{\sim}{E}\left[\left(\sum_{\underset{\sim}{s}^{\Gamma}_{*}\in\partial_i} 2^{-n_o R(\Gamma^*-m)/(1+\nu)}\left[\frac{w(\underset{\sim}{y}^{\Gamma^*}/\underset{\sim}{x}^{\Gamma^*}(\underset{\sim}{s}^{\Gamma}_{*}))\,w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{\Gamma^*})\,w(\underset{\sim}{y}^{m}/\underset{\sim}{x}^{m})}\right]^{1/(1+\upsilon)}\right)^{\upsilon}\right] \qquad (26)$$

for $m < \Gamma$

where the expectation is with respect to the symbol selection along the paths $\underset{\sim}{s}^{\Gamma}_{*}$. But for $0 \leq \nu \leq 1$, $E[|X|^{\nu}] \leq [E|X|]^{\nu}$, so since there are less than $2^{(\Gamma-i)n_o R}$ paths $\underset{\sim}{s}^{\Gamma}_{*}$ in the set ∂_i, the expression (26) may be upper bounded by

$$(\exp_2 \upsilon n_o R[(\Gamma-i) - (\Gamma^*-m)/(1+\upsilon)]) \times$$

$$\times\left\{\sum_{\underset{\sim}{y}^{\Gamma^*-i}} w_{\underset{\sim}{r}}(\underset{\sim}{y}^{\Gamma^*-i})\left(\sum_{\underset{\sim}{x}^{m-i}}\left[\frac{w(\underset{\sim}{y}^{m-i}/\underset{\sim}{x}^{m-i})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{m-i})}\right]^{1/(1+\upsilon)} r(\underset{\sim}{x}^{m-i})\right)\right\}\times$$

$$\times\left(\sum_{\underset{\sim}{x}^{\Gamma^*-i}_{*}}\left[\frac{w(\underset{\sim}{y}^{\Gamma^*-i}/\underset{\sim}{x}^{\Gamma^*-i}_{*})}{w_{\underset{\sim}{r}}(\underset{\sim}{y}^{\Gamma^*-i})}\right]^{1/(1+\upsilon)} r(\underset{\sim}{x}^{\Gamma^*-i}_{*})\right)^{\upsilon} \qquad (27)$$

if $0 \leq \upsilon \leq 1$. By Holder's inequality (p. 512, reference 1),

$$\underset{\sim}{E}[|XY|] \leq (E|X|^{1+\upsilon})^{\frac{1}{1+\upsilon}}\,(E|Y|^{\frac{1+\upsilon}{\upsilon}})^{\frac{\upsilon}{1+\upsilon}} \qquad (28)$$

$$\left[\sum_{\chi^{\Gamma^*-i}} w_{\underset{\sim}{r}}(\chi^{\Gamma^*-i})\left(\sum_{\underset{\sim}{x}^{m-i}}\left[\frac{w(\chi^{m-i}/\underset{\sim}{x}^{m-i})}{w_{\underset{\sim}{r}}(\chi^{m-i})}\right]^{\frac{1}{1+\nu}} r(\underset{\sim}{x}^{m-i})\right)^{1+\nu}\right]^{\frac{1}{1+\nu}}$$

$$\times\left[\sum_{\chi^{\Gamma-i}} w_{\underset{\sim}{r}}(\chi^{\Gamma^*-i})\left(\sum_{\underset{\sim}{x}_*^{\Gamma-i}}\left[\frac{w(\chi^{\Gamma^*-i}/\underset{\sim}{x}_*^{\Gamma^*-i})}{w_{\underset{\sim}{r}}(\chi^{\Gamma^*-i})}\right]^{\frac{1}{1+\nu}} r(\underset{\sim}{x}_*^{\Gamma^*-i})\right)^{1+\nu}\right]^{\frac{\nu}{1+\nu}} =$$

$$= \exp_2 -\frac{1}{1+\nu}\left[(m-i)\,E_o(\nu,\underset{\sim}{r}) + \nu(\Gamma^*-i)\,E_o(\nu,\underset{\sim}{r})\right] \tag{29}$$

for $m < \Gamma$

where we made use of definition (22). It is obvious that for $m = \Gamma$ all of the m's in (29) must be replaced by Γ^*. Using (12), (25), (27), and (29) we thus get the bound

$$\underset{\sim}{E}[P_e(i)] \le \{\exp_2 -\frac{\nu}{1+\nu} t\,[E_o(\nu,\underset{\sim}{r}) + n_o R]\} \times$$

$$\times\left(\exp_2\left[-\frac{\nu}{1+\nu}(\Gamma-i)\,[E_o(\nu,\underset{\sim}{r}) - \nu n_o R]\right]\right)\left(\sum_{m=1}^{\Gamma-1-i}\exp_2 -\frac{m}{1+\nu}[E_o(\nu,\underset{\sim}{r}) - \nu n_o R]\right)$$

$$+ \exp_2 -\{(\Gamma-i)[E_o(\nu,\underset{\sim}{r}) - \nu n_o R] + t\,E_o(\nu,\underset{\sim}{r})\} \tag{30}$$

where $0 \le \nu \le 1$

The righthand side of (30) again involves a geometrical series that is convergent provided $\nu \in [0,1]$ can be chosen so that

$$E_o(\nu,\underset{\sim}{r}) > \nu n_o R \tag{31}$$

It is easily shown that $E_o(\nu,\underset{\sim}{r})$ is a positive, increasing, concave function of ν for $\nu \ge 0$. Therefore, if condition (24) is satisfied then so can (31) be. The distribution $r(x)$ which is used to select the tree code should thus be chosen so as to maximize $E_o(1,\underset{\sim}{r})$. Let $r^*(x)$ be the maximizing distribution, choose $\nu = 1$, and let

$$A(R) \triangleq \sum_{m=1}^{\infty} 2^{-\frac{m}{2}[E_o(1,\underset{\sim}{r}^*) - n_o R]}$$

$$B(R) \triangleq \sum_{j=1}^{\infty} 2^{-j[E_o(1,\underset{\sim}{r}^*) - n_o R]} \tag{32}$$

Then plugging (30) into (13), we get that the average probability of error $\underset{\sim}{E}[P_e]$ (over the code ensemble) is bounded by

$$\underset{\sim}{E}[P_e] \le A(R)^2\, 2^{-\frac{1}{2}t\,[E_o(1,\underset{\sim}{r}^*) + n_o R]} + B(R)\, 2^{-t\,E_o(1,\underset{\sim}{r}^*)} \tag{33}$$

and the average decoding effort $\underset{\sim}{E}[N_i]$ per decoded digit (see (23)) is bounded by

$$\underset{\sim}{E}[N_i] \le 1 + A^2(R) \tag{34}$$

where A(R) and B(R) are finite provided the transmission rate R is less than the quantity

$$R_{comp} \triangleq \frac{1}{n_o} \max_{\underset{\sim}{r}} E_o(1,\underset{\sim}{r}) = \frac{1}{n_o} E_o(1,\underset{\sim}{r}^*) \tag{35}$$

The first exponent on the righthand side of (33) increases with decreasing R, but obviously the probability of error must decrease. Consequently for $R < R_{comp}$,

$$\underset{\sim}{E}[P_e] \le [A(R)^2 + B(R)^2]\, 2^{-t\,R_{comp}} \tag{36}$$

It is easy to show (see Lemma 10.9 of reference 1) that there will exist at least one code for which simultaneously

$$P_e \le 2\,\underset{\sim}{E}[P_e]$$
$$\bar{N}_i \le 2\,\underset{\sim}{E}[N_i] \tag{37}$$

Actually, the reader may be assured that almost all codes selected at random will satisfy (37).

Close examination of (30) will show that the probability of error will decrease exponentially with t even when the rate satisfies $R_{comp} \le R < C$, where C is the channel capacity. However, the geometric series in (23) will not converge, and in fact if can be shown[8] that $\bar{N}_i$ will increase exponentially with Γ if $R > R_{comp}$. So for large block sizes Γ (which are needed to keep the rate loss factor $t/(\Gamma+t)$ small) one would ordinarily not attempt to use sequential decoding at rates that exceed R_{comp}.

5. PRACTICAL CONSIDERATIONS CONNECTED WITH THE STACK DECODING ALGORITHM AND SOME REFINEMENTS

As described at the end of Section 3, every decoding step of the tree search algorithm would involve the replacement of the top likelihood in the stack by d new likelihoods. This means a net growth of the stack size by (d-1) entries per decoding step. This unwanted dependence on d is entirely unnecessary, as follows from the observation (due to J. Cocke) that the path corresponding to the $(j+1)^{th}$ branch (in order of branch likelihood value) leaving a particular node can reach the top of the stack only after the path corresponding to the j^{th} branch did. Hence, we can modify the algorithm to limit the stack growth to at most one entry per decoding step regardless of the size of d:

1. With each path likelihood $L(\underset{\sim}{s}^i)$ store also the order j (by size) of the likelihood $\lambda^j = \lambda_i(\underset{\sim}{s}^i)$ (see definition (11)) of the last branch s_i of the path [where $L(\underset{\sim}{s}^i) = L(\underset{\sim}{s}^{i-1}) + \lambda^j$, $\underset{\sim}{s}^i = (\underset{\sim}{s}^{i-1}, s_i)$ and $\lambda^1 \geq \lambda^2 \geq \ldots \geq \lambda^d$].
2. If $L(\underset{\sim}{s}^i)$ is found at the top of the stack, replace it by the likelihood $L(\underset{\sim}{s}^{i+1})$, where s_{i+1} has the largest likelihood of all branches leaving $\underset{\sim}{s}^i$. If the order j of the likelihood of the branch s_i is less than d, insert into the stack also the likelihood $L(\underset{\sim}{s}^i_*) = L(\underset{\sim}{s}^{i-1}) + \lambda^{j+1}$ corresponding to the path through the (j+1) order branch leaving the terminal node of $\underset{\sim}{s}^{i-1}$. If j=d, do not insert any additional path.

With the above modification it becomes advantageous (in terms of decoding speed and stack sizeeconomy) to make the number of branches leaving a node as large as possible, provided their ordering in terms of likelihood value can be accomplished by a table look-up rather than by a direct computation followed by a comparison. If the tree code used has a convolutional structure (see section 10.12 of reference 1) it is possible to construct such tables, and the size of d is determined by the available storage size.

Next, let us consider the problem of maintaining the stack in proper likelihood order. Obviously, running down the stack, comparing the entries to the likelihood to be inserted, and then making space for the new entry when its location has been found is too unwieldy a method. Besides, the insertion

time would vary as a function of the stack content. It is better to establish equivalence classes for likelihoods (e. g. all likelihoods with the same integral part belong to the same class) and to provide corresponding class buckets for insertion of new entries. The buckets are then ordered and a decoding step consists of selecting an arbitrary path of the top bucket, computing the likelihood of its extensions, computing the class membership of these likelihoods, and inserting the corresponding paths into the appropriate buckets. To be more specific, we will describe the system used by the author during simulation.

There are two sets of reserved storage locations, the auxiliary stack and the stack itself, referred to by indices $\ell \in \{-K, -k+1, \ldots, 0, 1, \ldots, J\}$ and $b \in \{1, 2, \ldots, M\}$, where K and J are chosen so that the likelihood values of paths visited during the decoding process lie between -K and J with sufficiently large probability. M is the size of the stack, each entry of which consists of three parameters: S(g) -the path specification, L(g) - the likelihood value, and P(g) - a pointer to be described. The stack is filled sequentially as follows: the first path is put into location 1 (this path is then on top of the stack). Whenever a path is extended, its parameters are replaced by the corresponding parameters of the extended path. If an additional entry into the stack is made (this happens unless the last branch of the replaced path had order d) it is placed into the first unfilled location assuming there is one (if the stack is full, the bottom-of-the-stack path is replaced, as discussed below). Let $G(\ell)$ be the entry at location ℓ of the auxiliary stack. Its value is equal to the stack address of the last entered path whose likelihood has an integral part equal to ℓ (if no such path exists in the stack, $G(\ell) = 0$). The pointer P(g) is equal to the stack address of the previously entered path whose likelihood has an integral part equal to the integral part of L(g) (if no such path exists, P(g) = 0).

The appropriate parameter values are maintained in the two storages as follows: suppose the path at location b of the stack is to be extended (and therefore eliminated from the stack) and suppose ℓ equals the integral part of the likelihood L(g). Then the entry $G(\ell)$ of the auxiliary stack will be set equal to the pointer P(g). Next, suppose the path S' of likelihood L' is to be inserted into location g' of the stack. If ℓ' is the integral part of L', then $P(g')$ is set equal to the current entry $G(\ell')$ of the auxiliary stack, whereupon $G(\ell')$ is reset to equal g'. Finally, $L(g')$ is set equal to L' and $S(g')$ to S'.

Table I illustrates the stack maintenance procedure in the case of the likelihood situation of Fig. 4.

The auxiliary stack entries $G(\ell)$ are used to find the top-of-the-stack path to be extended and the bottom-of-the-stack path to be eliminated when a new entry is to be made into a full stack. The address of the top of the stack is $G(\ell^*)$ where ℓ^* is such that $G(\ell) = 0$ for $\ell = \ell^*+1, \ldots J$ and $G(\ell^*) \neq 0$. When the stack is full, the new entry is put into location $g^+ = G(\ell^*)$ where $G(\ell) = 0$ for $\ell = -K, \ldots, \ell^+-1$ and $G(\ell^+) \neq 0$. Before this is done, $G(\ell^+)$ is set equal to $P(g^+)$.

As just stated, when a new entry is to be placed into a full stack, an old entry must be purged from the latter. The worry is that the purged entry may correspond to the true path in which case a decoding error will necessarily result unless other measures are taken. One possibility, described in some detail below, is to switch into a Fano decoding[3] mode (for description of Fano's sequential decoding algorithm, see section 10.4 of reference 1). Let T be the integral part of the largest likelihood corresponding to a path that was purged from the stack. We will say that a stack overflow took place if the path at the top of the stack has likelihood $L < T$. Obviously, when the stack overflows it is futile to continue operating in the old way. If another operational mode is not available, one might as well stop decoding. In the last Section we shall present some simulation results pertaining to this problem.

Next, let us consider how a path is to be specified in the stack. If the tree code is convolutional (see Section 10.12 of reference 1) then at a minimum the stack parameters S(b) must characterize the tree depth i of the path and the sequence of υ last message digits $s_{i-\upsilon+1}, \ldots, s_i$, where υ is greater than or equal to the constraint length of the code. If υ is less than $\Gamma + t$, the block length of the code, then a way must be found to determine the decoded path. It would be natural to release to the user the digit $s_{i-\upsilon+1}$ whenever the tree depth i of the path at the top of the stack exceeds the depth of all the previously extended paths. In order to keep the probability of a wrongly released digit small, υ might have to be as large as three times the constraint length. This might make the stack storage unacceptably large and so a more subtle method of path specification might have to be devised.

A possible solution is to maintain in the storage a "map" of the paths contained in the stack. This could be done as follows. An integer k is chosen and map storage space is

allocated whose entries $V^*(j)$ and $Q^*(j)$ at location j represent a sequence of k message digits, and a pointer, respectively. The stack parameter S(g) itself consists of three parts, V(g) capable of representing k-1 message digits, I(g) representing the path depth, and Q(g), a pointer. At the beginning, $V^*(j) = Q^*(j) = V(g) = Q(g) = I(g) = 0$ for all j and g. As long as the path depth $I(g) < k$, then $Q(g) = 0$ and $V(g) = s_1, \ldots, s_{I(g)}$ the message sequence. As soon as for the first time a path $\underset{\sim}{s}^{k-1}$ is to be extended to $(\underset{\sim}{s}^{k-1}, s_k)$, we set $V^*(1) = (\underset{\sim}{s}^{k-1}, s_k)$, V(g) = 0, I(g) = k, and the pointer Q(g) is set equal to 1, thus pointing to the first map location. In general, whenever path extension occurs from a path $\underset{\sim}{s}^{i-1}$ to $\underset{\sim}{s}^{i}$ where i is not a multiple of k, neither the pointers Q(g) nor $Q^*(j)$ are changed, no new entry in the map is made, I(g) is set to i, and s_i is added to the content of V(g). Suppose that $\underset{\sim}{i}$ is a multiple of k and that j is the first free location in the map. Then the pointer $Q^*(j)$ is set equal to the old pointer value Q(g), $V^*(j)$ is set equal to $s_{i-k+1}, \ldots, s_i$, Q(g) is made to point to the map location j, and we make I(g) = i and V(g) = 0. It is obvious that in this way the entire path corresponding to any stack entry g is kept available in the storage. In fact, the path digits s_i, s_{i-1}, $\ldots, s_1$ are represented in the registers V(g), $V^*(j_1), \ldots, V^*(j_t)$ where $j_1 = Q(g)$, $j_2 = Q^*(j_1), \ldots, j_t = Q^*(j_{t-1})$, and $Q^*(j_t) = 0$.

If it is desired to purge the map of entries made unnecessary by path purges of the stack, this may best be accomplished by adding another register $C^*(j)$ at the various map locations. $C^*(j)$ will at all times be equal to the number $v (\leq d^k)$ of pointers pointing to the map location j. Suppose the entry g is to be purged from the stack and $j_1 = Q(g)$. Then the value of $C^*(j_1)$ is lowered by 1. If the new value is not equal to 0 nothing further is done, if it is equal to 0 and $j_2 = Q^*(j_1)$ then $C^*(j_2)$ is lowered by 1 and the j_1 map location is made available to a pool for new refilling. The new value of $C^*(j_2)$ is similarly compared to 0, and if it is 0 then $C^*(j_3)$ $[j_3 = Q^*(j_2)]$ is in turn lowered by 1 and the j_2 map location becomes available, etc.

6. EVALUATION, COMPARISONS, AND CONCLUSIONS

The author has not yet fully evaluated the performance of the stack algorithm, but he can present some results of computer simulation and compare these with the published performance of the Fano sequential decoding algorithm.

We have run simulations of the performance of a rate $R = \frac{1}{2}$ code when used over a binary symmetric channel whose crossover probability p is allowed to vary. In our experiment, the block length $\Gamma = 1000$, and the length of the last level branches was t = 26 symbols. The results of Figs. 7 and 8 are based on 1000 such blocks run for each of the different values of p. We have run the decoding as described in Section 3 with two branches leaving each node, and also as described in Section 5, with sixteen branches leaving each node. The solid line in Fig. 7 indicates the average number of stack algorithm decoding steps necessary to decode one tree branch. It is remarkable that at all rates tested, this number is virtually the same whether the branch represents one (two branches per node) or four (sixteen branches per node) binary message digits. The interrupted line plots the same quantity for the Fano decoding algorithm (the data is taken from Fig. 6.50 of the text by Wozencraft and Jacobs[9]). It is seen that at R_{comp} the stack algorithm enjoys better than a sixfold advantage. Figure 8 plots the maximum position of the true path in the stack over the 1000 blocks decoded (i.e. if the stack used were longer than the number given, the true path would never have been eliminated from it). The solid line contains the data for the two-branch-per-node set up, the interrupted line for the sixteen-branch-per-node set up. It is clear that as expected the second set up does allow for substantial economies in needed stack depth.

It ought to be stressed that Fig. 7 does not present the entire story and is actually unfavorable to the stack decoding algorithm. Namely, when decoding in real time with a fixed maximum delay imposed on the release of decoded information to the user (relative to the time of reception), it is the peak demands on computation that ought to be compared. Since the advantage of the stack algorithm grows substantially as the rate approaches R_{comp}, it ought to be expected that this advantage will be even more pronounced during periods of high channel noise and consequent high computational demand.

Because of the above considerations the decoder <u>speed factor</u> (maximal number of decoding steps performable in the

time it takes to receive one branch) must exceed substantially the averages plotted in Fig. 7. This is especially true for the Fano decoder that will be idle during the low noise time intervals when it is "caught up" with the received signal (in such situations it can perform only one step per received branch interval, regardless of what its speed factor is!). The stack decoder need never be idle. If the length of the path on the top of the stack is equal to the length of the received sequence, the decoder may profitably extend the highest situated of those paths in the stack that are shorter than the top one. This apparently premature work costs nothing, and will not have to be done later should the channel noise increase.

The complete mutual independence of the stack ordering and path extending portions of the algorithm is also worthy of note. These two functions can be performed in parallel by different (but communicating) machines. The stack need not even be in order--only the top path must be available for extension. To achieve further speed-up at the cost of higher decoder complexity one might conceivably use several path extending machines simultaneously, the first one working on the top path in the stack, the second one on the second path, etc.

We will conclude this paper with a description of how stack overflows can be handled by a temporary switch to a Fano decoding mode. Thus, it will be seen that even relatively short stacks capable of storing only several hundred paths can be used to speed up the decoding process, or from the opposite point of view, that catastrophe can be avoided when stack overflow takes place. To keep the following discussion brief, it will be assumed that the reader is thoroughly familiar with some version of the Fano decoding algorithm (e.g. the one presented in section 10.4 of reference 1).

We have already discussed in Section 5 the phenomenon of stack overflow. We will recapitulate part of that discussion and make a minor modification in the stack insertion part of the algorithm. As before, let T be the integral part of the largest path likelihood that was ever purged from the stack. Obviously, T is a monotonically non-decreasing function of time which might as well be defined equal to $-\infty$ until the first path is purged from the stack. Let us never insert a path into the stack whose likelihood is less than the current value of T. With this change in the insertion procedure, stack overflow is equivalent to finding the stack empty and therefore not being

able to continue to carry out the algorithm. Let $\underset{\sim}{s}^i$ be the last path extended before stack overflow took place. Then obviously $L(\underset{\sim}{s}^i) \geq T$ and $L(\underset{\sim}{s}^i, s_{i+1}) < T$ for all branches s_{i+1} leaving the node $\underset{\sim}{s}^i$. First, the decoder backs up to the nearest preceeding node $\underset{\sim}{s}^j$ [$j \leq i$, $\underset{\sim}{s}^i = (\underset{\sim}{s}^j, s_{j+1}, \ldots, s_i)$] whose immediate predecessor node $\underset{\sim}{s}^{j-1}$ has likelihood $L(\underset{\sim}{s}^{j-1}) < T$. Next, the decoder is switched into the Fano forward mode, setting its cumulative threshold T_o equal to $T-\tau$, where τ is the Fano decoder threshold quantum. Decoding continues in the Fano mode with T_o varying as needed until such time as the decoder arrives for the first time at some node $\underset{\sim}{s}^k$ at which the current value of T_o is to be raised (until this time the variation in T_o was downward only). At this moment $\underset{\sim}{s}^k$ is inserted into the stack, T is set equal to $T_o + \tau$ (note that $L(\underset{\sim}{s}^k) \geq T_o + \tau$) and decoder operation resumes in the stack mode. Obviously, if $\underset{\sim}{s}^k$ is not the true path then with high probability all of the paths leaving it will have likelihood less than the new value of T. In such a case the stack will again be found to be empty, and the decoder will switch back to the Fano mode as described above, etc. It is important to note that this switchback will occur after fewer decoding steps have been completed than those that would have been necessary for the Fano decoder to lower its threshold back from $T_o + \tau$. Thus, our strategy wastes no steps by a possibly premature switch into the stack decoding mode.

ACKNOWLEDGMENT

The author wishes to thank Dr. John Cocke for his inspiration and many valuable discussions and suggestions.

REFERENCES

1. F. Jelinek: Probabilistic Information Theory, McGraw-Hill, New York, 1968.

2. J. M. Wozencraft: "Sequential Decoding for Reliable Communication", MIT-RLE Tech. Rept., TR325, 1957.

3. R. M. Fano: "A Heuristic Introduction to Probabilistic Decoding", IEEE Trans. on Information Theory, IT-9, April, 1963.

4. M. S. Pinsker: "O Sloznosti Dekodirovaniia", Problemy Peredachi Informatsii, 1(1): 113-116 (1965).

5. D. Falconer: "A Hybrid Sequential and Algebraic Decoding Scheme," Sc. D. thesis, Dept. of Elec. Eng., M. I. T., 1966.

6. F. L. Huband and F. Jelinek: "Practical Sequential Decoding and a Simple Hybrid Scheme", Hawaii Int. Conf. on System Sciences, January , 1968.

7. G. D. Forney, Jr.: "Generalized Minimum Distance Decoding", IEEE Trans. on Information Theory, IT-12, (2), April, 1966.

8. I. M. Jacobs and E. Berlekamp: "A Lower Bound to the Distribution of Computation for Sequential Decoding", IEEE Trans. Inform. Theory, IT-13(2): (April, 1967).

9. J. M. Wozencraft and I. M. Jacobs: Principles of Communication Engineering, John Wiley & Sons, Inc., New York, 1965.

Table 1. The contents of the stack and the auxiliary stack during the search of the tree of Figure 4.

STEP 1

b	S	L	P	ℓ	g
1	0	-6	0		
2	1	1	0		
				1	2
				0	0
				-6	1

STEP 2

b	S	L	P	ℓ	g
1	0	-6	0		
2	10	-3	0		
3	11	-3	2		
				-3	3
				-6	1

STEP 4

b	S	L	P	ℓ	g
1	0	-6	0		
2	100	-7	4		
3	110	-7	0		
4	111	-7	3		
5	101	-7	2		
				-5	0
				-6	1
				-7	5
				-8	0

STEP 6

b	S	L	P	ℓ	g
1	000	-11	0		
2	100	- 7	4		
3	110	- 7	0		
4	111	- 7	3		
5	101	- 7	2		
6	01	-12	0		
7	001	- 4	0		
				-4	7
				-7	5
				-11	1
				-12	6

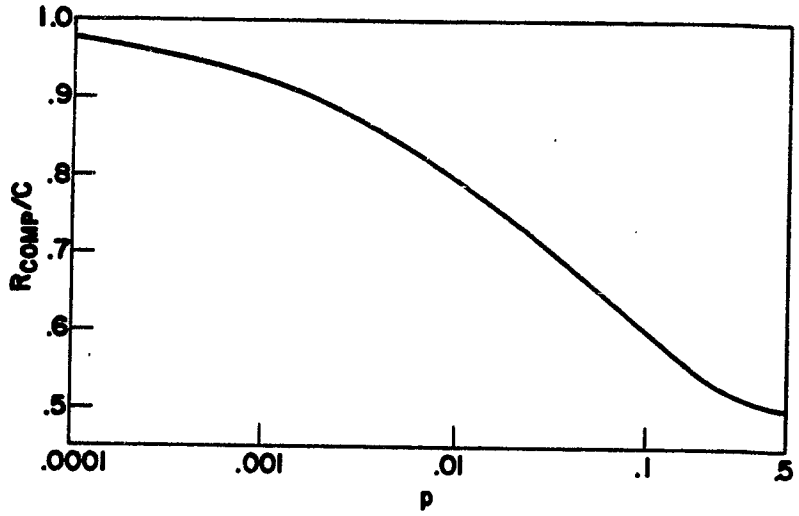

Figure 1. The graph of R_{comp}/C as a fucntion of the crossover probability p of a binary symmetric channel.

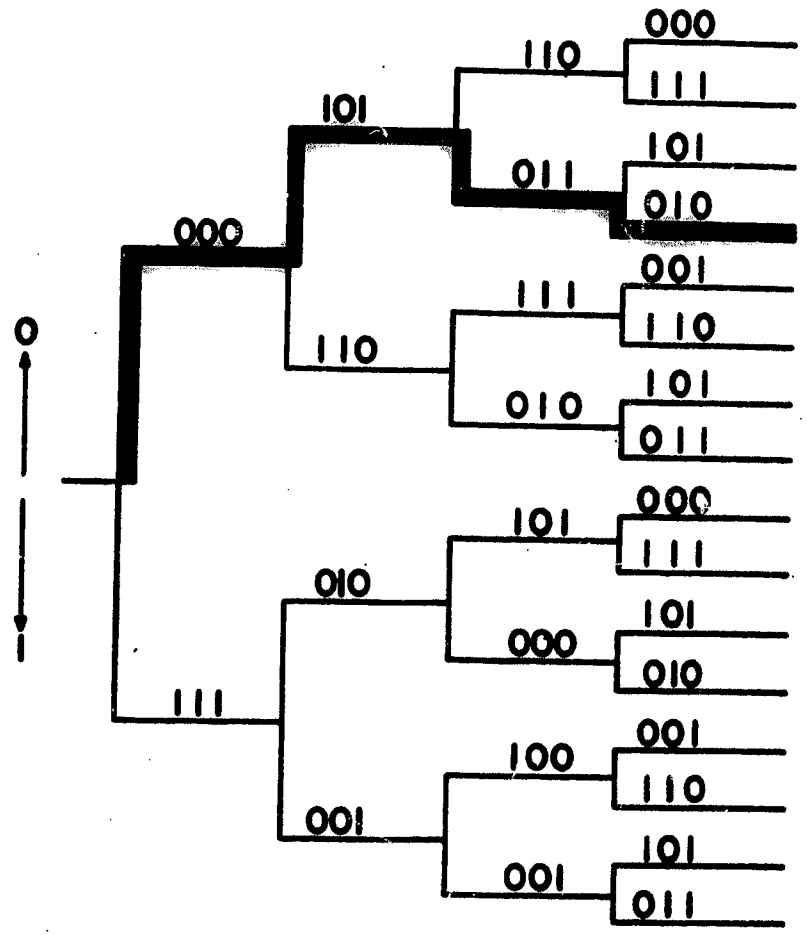

Figure 2. Example of a binary tree code of rate R = 1/3.

Figure 3. A block diagram of a communication system that encodes information blocks of length Γ into codewords of length n.

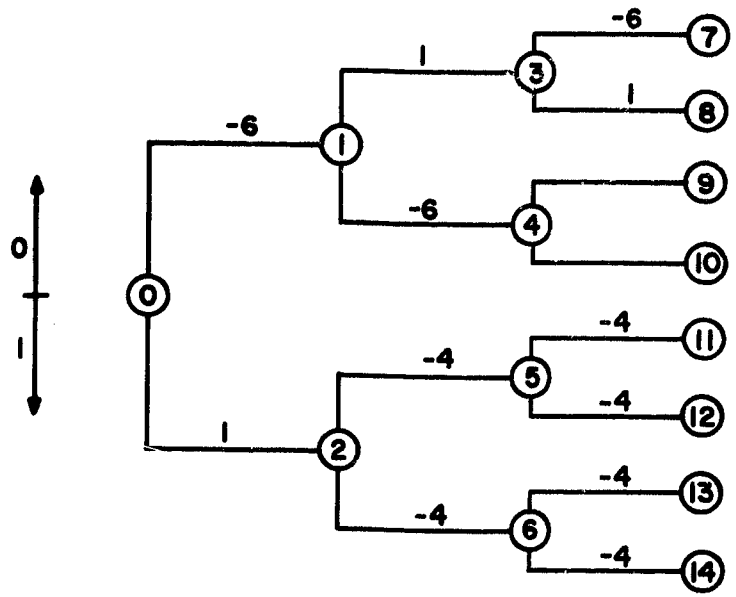

Figure 4. An example of likelihood value assignment to tree branches induced by a code and a recieved sequence.

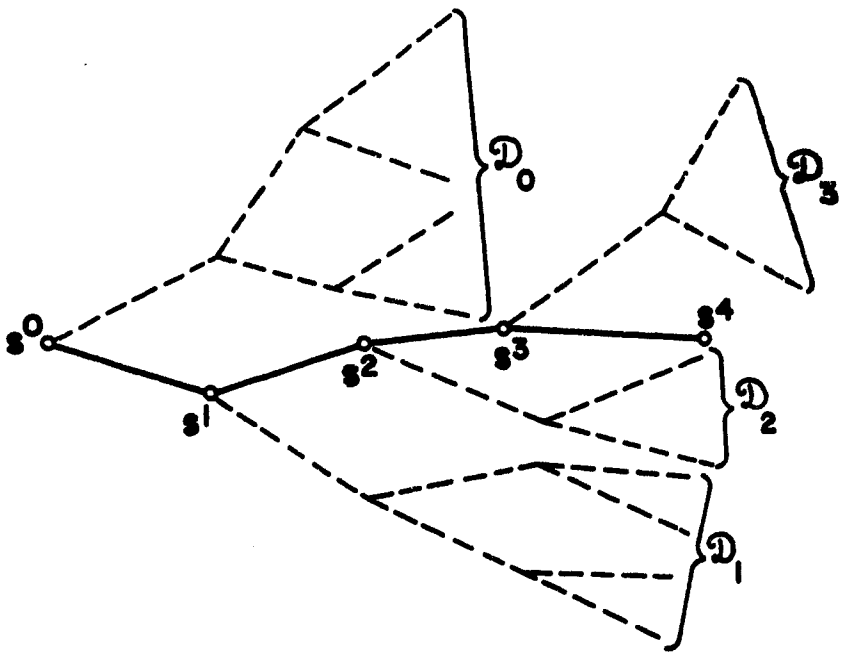

Figure 5. The $\mathcal{D}_i$ classification of nodes of a binary tree relative to the transmitted message $(s_1, s_2, s_3, s_4, \ldots)$.

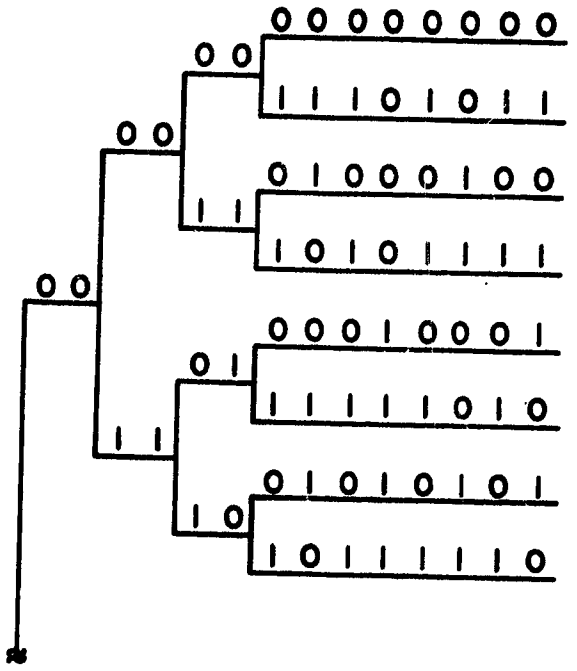

Figure 6. Portion of the last levels of a tree whose last branches have length $(t + 1) = 4$ times that of regular branches.

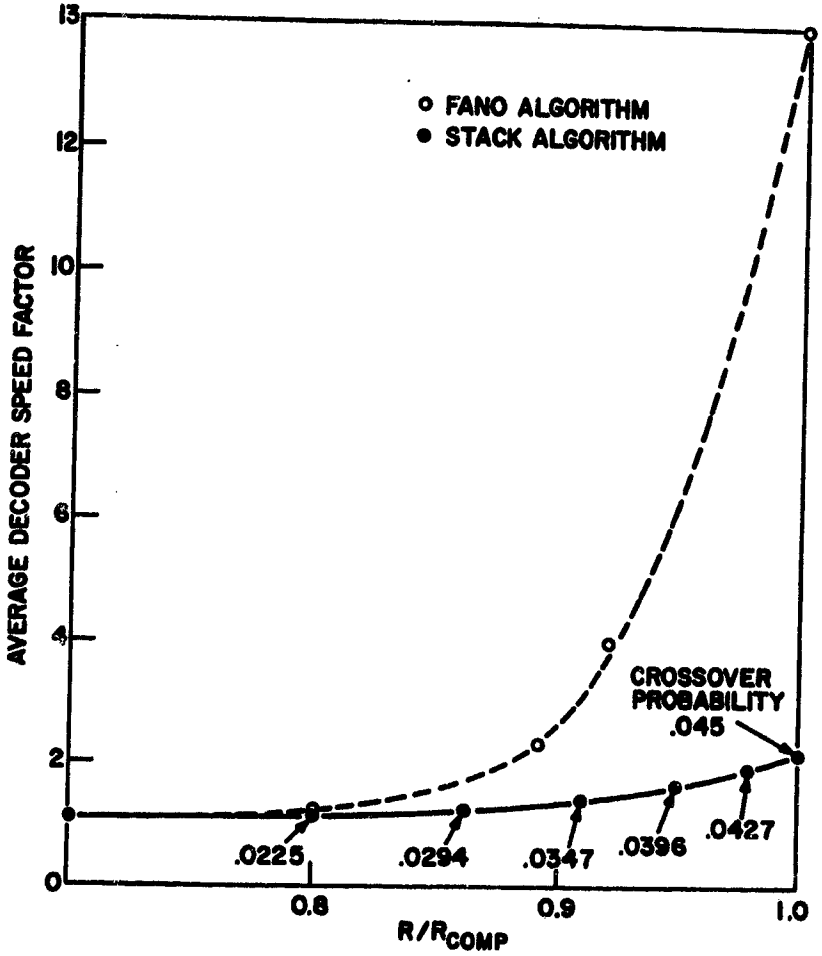

Figure 7. Average number of decoding steps necessary to decode a tree branch using the stack algorithm (solid line) and the Fano algorithm (interrupted line) as a function of the rate-to-R_{comp} ratio. Coding rate is held constant at $R = \frac{1}{2}$ but the crossover probabilities of the binary symmetric channel vary as indicated. Stack algorithm results are based to 1000 runs of binary information blocks of length 1000.

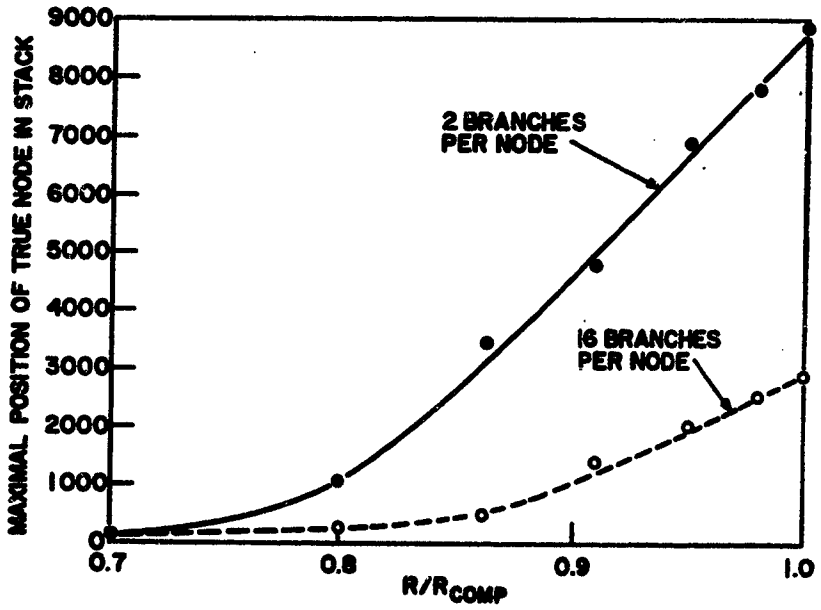

Figure 8. Plot of the maximal position of the true node in the stack obtained from 1000 runs of binary information blocks of length 1000. Solid line applies when the tree has two branches leaving each node and the stack insertion procedure is that described in Section 3. Interrupted line applies when the tree has sixteen branches leaving each node and the insertion procedure is that of Section 5.