Technische Universität München
Coding for Communications and Data Storage
Prof. Dr.-Ing. Antonia Wachter-Zeh

Master's Thesis

Sequential Decoding of Convoutional Codes for Insertion/Deletion Channels

Vorgelegt von:
Anisha Banerjee

München, March 2021

Betreut von:
M.Sc. Andreas Lenz

Master's Thesis am
Coding for Communications and Data Storage (COD)
der Technischen Universität München (TUM)
Titel : Sequential Decoding of Convoutional Codes for Insertion/Deletion Channels
Autor : Anisha Banerjee

Anisha Banerjee
Heiglhofstr. 66
81377 München
anisha.banerjee@mytum.de

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

München, 03.03.2021

Ort, Datum (Anisha Banerjee)

Contents

List of Figures

List of Tables

Abstract

Certain communication channels can be vulnerable to insertion, deletion and substitution errors. Under such conditions, convolutional codes may be used to increase reliability of data transmission. The optimal decoding procedure for this class of error-correcting codes is the Viterbi algorithm. This method however, demands significant memory and computational effort even for moderate constraint lengths. It is hence worthwhile to explore alternative decoding strategies that only examine the relevant parts of a trellis. To this end, we explore the viability of a sequential decoder. By disregarding the improbable paths in the trellis altogether, it stands to greatly reduce decoding complexity, especially under low-noise environments. In this work, we compare its error-correcting performance and computational complexity with the Viterbi algorithm. While it does offer huge savings in computational effort albeit at poorer error rates, these advantages disappear at higher noise levels.

1 Introduction

Convolutional codes constitute a class of error-correcting codes that incorporates memory in its encoding process, as opposed to the block coding technique. They have gained enormous favor in the areas of wireless, broadcast and space communications, and the most popular choice of decoder is the Viterbi algorithm. This method essentially provides a maximum-likelihood estimate of the transmitted sequence and can be implemented quite easily.

For codes with longer constraint lengths however, the high computational complexity of the Viterbi decoder renders it impractical, thereby prompting a second look at an alternative strategy, namely sequential decoding. This technique was originally proposed by Wozencraft [**?** **?**] in 1957, and subsequently improved by Fano [**?**]. It garners significant interest since its complexity is virtually independent of encoder memory. Furthermore, a sequential decoder only examines those sequences that seem likely to have been transmitted, unlike the Viterbi decoder which always assesses all possibilities, regardless of noise levels. Though this leads to a suboptimal decoding accuracy, it also potentially implies that sequential decoders can significantly reduce computational effort, especially in environments with low error probabilities.

It is vital to expand this study of decoders for convolutional codes to a more general channel model, that allows insertion and deletion errors, in addition to substitutions. Such errors may be induced due to improper transmitter-receiver synchronization, and often occur in data storage and networking channels. Prior work done in [**?** **?** **?**] facilitates this extension.

The aim of this work is to extend the application of sequential decoders to channels that are susceptible to insertion, deletion and substitution errors. Among the several variants of sequential decoding, focus is confined to Fano's algorithm. We begin with a brief description of the channel model, followed by a discussion on the decoding algorithm. In order to allow the use of Fano's decoder for this particular channel, we also derive a new decoding metric that appropriately generalizes the original metric proposed by Fano. Subsequently we compare its performance with the Viterbi decoder in regard to error rates and computational effort. An analytical assessment of the algorithm's average complexity is also performed.

2 Binary Substitution, Insertion and Deletion Channel

The channel model adopted for this work is the Binary Substitution, Insertion and Deletion (BSID) channel defined in [**?** **?**]. Henceforth, we view it as a state machine, characterized by three parameters: P_i, P_d and P_s, which refer to the insertion, deletion and substitution probabilities respectively.

2.1 Channel Model

Let $\boldsymbol{x} = (x_1, \dots x_T) \in \mathbb{F}_2^T$ be the encoded frame awaiting transmission, and let S_i denote the channel state reached after i uses. Under this construct, the channel starts from the initial state S_0, and after completing transmission, terminates at state S_T. When in an intermediate state S_i, any of the following three events is possible, as depicted in Fig. 2.1:

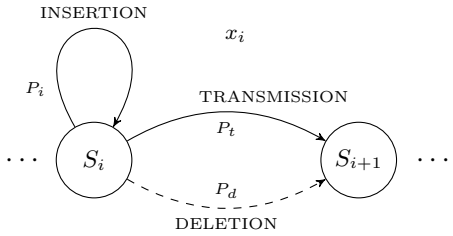


Figure 2.1: Transitions allowed from a state

1. Insertion: A random bit y in inserted into the received stream. This is represented as a self-transition back to state S_i, with a transition likelihood $p(S_i|S_i) = P_i$, and an observation likelihood $q(y|S_i \to S_i) = 1/2$.

2. Transmission: The bit y is received due to transmission of bit x_{i+1} and the channel transitions into state S_{i+1} . The probability of such an output-producing transition is given by $p(S_{i+1}, y|S_i, x_{i+1}) = P_t q(y|x_{i+1})$, where $P_t = 1 - P_i - P_d$ is the transition

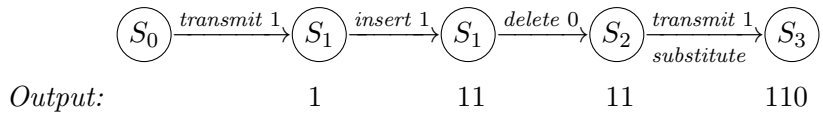
likelihood while the observation likelihood is expressed as:

$$q(y|x_{i+1}) = \begin{cases} 1 - P_s, & \text{if } y = x_{i+1} \\ P_s, & \text{else} \end{cases}$$

3. Deletion: Bit x_{i+1} is deleted and no output is produced. This corresponds to a null-output transition into state S_{i+1}, that occurs with probability $p(S_{i+1}, y = \phi|S_i, x_{i+1}) = P_d$.

Additionally, the representation in Fig. 2.1 indicates that the number of consecutive insertions in the channel output is not constrained.

Example 2.1.1. *Consider a channel input $\boldsymbol{x} = (1, 0, 1)$. A possible sequence of events might be:*



Thus the resulting received sequence is $\boldsymbol{y} = (1, 1, 0)$, and probability of the channel traversing this path, is given by:

$$\begin{aligned} P(\boldsymbol{y}, \{S_0, S_1, S_1, S_2, S_3\}|\boldsymbol{x}) &= P_t(1 - P_s) \cdot \frac{P_i}{2} \cdot P_d \cdot P_t P_s \\ &= \frac{1}{2} P_t^2 P_d P_i P_s (1 - P_s) \end{aligned}$$

It is worth noting that the final state S_3 has no outgoing edges, implying that no output-producing transitions, including insertions, are allowed after transmission of $\boldsymbol{x}$ has ended.

2.2 Likelihood Metric

To compute $P(\boldsymbol{y}|\boldsymbol{x})$, i.e the likelihood of receiving $\boldsymbol{y} = (y_1, \ldots, y_N) = y_1^N$, given that the sequence transmitted is $\boldsymbol{x} = (x_1, \ldots x_T) = x_1^T$, it is crucial to note that there are multiple paths from S_0 to S_T that the channel's state machine could have traversed, such that all of them led to the same received sequence, i.e $\boldsymbol{y}$. In the context of example 2.1, this means that given a transmitted sequence $\boldsymbol{x} = (1, 0, 1)$, $\boldsymbol{y} = (1, 1, 0)$ could also have been received due to the following state traversals:

- 

- 

- $(S_0) \xrightarrow{\text{transmit 1}} (S_1) \xrightarrow{\text{insert 1}} (S_1) \xrightarrow{\text{insert 0}} (S_1) \xrightarrow{\text{delete 0}} (S_2) \xrightarrow{\text{delete 1}} (S_3)$

Thus, in order to compute $P(\boldsymbol{y}|\boldsymbol{x})$, the probabilities of all such state traversals need to be accounted for, as follows:

$$\begin{aligned} P(\boldsymbol{y}|\boldsymbol{x}) &= P(\boldsymbol{y}, \{S_0, S_T\}|\boldsymbol{x}) \\ &= \sum_{S \in \mathbb{S}} P(\boldsymbol{y}, S|\boldsymbol{x}) \end{aligned} \tag{2.1}$$

where $\mathbb{S}$ denotes the set of all possible state sequences that start from S_0 and end at S_T, where T is the number of bits in the transmitted sequence $\boldsymbol{x}$. To simplify this computation, we adopt the algorithm proposed in [**?**]. This approach exploits the fact that state S_T can only be entered from the penultimate state S_{T-1}, either by transmission or deletion. Mathematically, this translates to:

$$\begin{aligned} P(\boldsymbol{y}_1^N|\boldsymbol{x}_1^T) &= P(\boldsymbol{y}_1^N, \{S_0, S_T\}|\boldsymbol{x}_1^T) \\ &= P_t q(y_N|x_T) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) + P_d P(\boldsymbol{y}_1^N, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \end{aligned} \tag{2.2}$$

Now, to compute $P(\boldsymbol{y}_1^n, \{S_0, S_m\}|\boldsymbol{x}_1^T)$ where $m < T$ and $n \leq N$, we note from Fig. 2.1 that unlike the terminal state S_T, an intermediate state S_m has three incoming edges, two of which originate from the preceding state S_{m-1}, while the third is a self-transition, denoting an insertion. Hence the corresponding recurrence relation would be:

$$\begin{aligned} P(\boldsymbol{y}_1^n, \{S_0, S_m\}|\boldsymbol{x}_1^T) &= P_t q(y_n|x_T) P(\boldsymbol{y}_1^{n-1}, \{S_0, S_{m-1}\}|\boldsymbol{x}_1^T) + P_d P(\boldsymbol{y}_1^n, \{S_0, S_{m-1}\}|\boldsymbol{x}_1^T) \\ &\quad + \frac{P_i}{2} P(\boldsymbol{y}_1^{n-1}, \{S_0, S_m\}|\boldsymbol{x}_1^T) \end{aligned} \tag{2.3}$$

For notational convenience, let

$$F_{m,n} = P(\boldsymbol{y}_1^n, \{S_0, S_m\}|\boldsymbol{x}_1^T)$$

for a specific $\boldsymbol{x} = \boldsymbol{x}_1^T$. Hence (2.2) and (2.3) can be restated as follows:

$$F_{m,n} = \begin{cases} P_t q(y_n|x_m) F_{m-1,n-1} + P_d F_{m-1,n}, & \text{if } m = T \\ P_t q(y_n|x_m) F_{m-1,n-1} + P_d F_{m-1,n} + \frac{P_i}{2} F_{m,n-1}, & \text{if } 0 \leq m < T \end{cases} \tag{2.4}$$

Thus, computing $P(\boldsymbol{y}|\boldsymbol{x})$ simply involves using the above recurrence relations, which can be visualized through the lattice grid illustrated in Fig. 2.2. The evaluations begin from the upper-left corner of the grid, which indicates that the channel is at its initial state S_0, and that no bits have been received. All of the vertices lying along the main diagonal

represent zero drift. The row indices of the lattice range from 0 to N, while the column

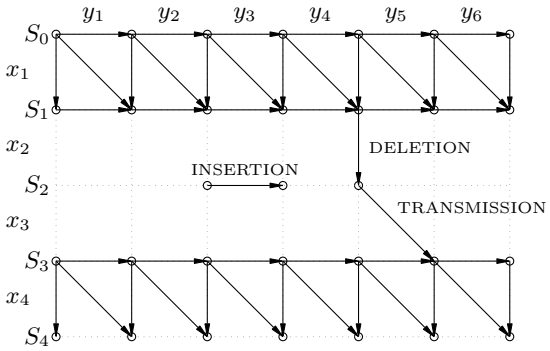


Figure 2.2: Lattice structure

indices range from 0 to T. Its horizontal edges representing insertions are of weight $\frac{1}{2}P_i$ while the vertical deletion edges have a weight of P_d. Finally, a diagonal 'transmission' edge entering node at (m, n) has weight $P_t q(y_n|x_m)$. Also observing that at start of transmission, channel is in state S_0 and received sequence is $\boldsymbol{y} = \emptyset$ with certainty, we define the following base cases:

$$F_{m,n} = \begin{cases} 0, & \text{if } m < 0 \text{ or } n < 0 \\ 1, & \text{if } m = n = 0 \end{cases} \tag{2.5}$$

In summary, the likelihood of receiving a vector $\boldsymbol{y} = y_1^N$ given that the bit sequence $\boldsymbol{x} = x_1^T$ was transmitted, is given by:

$$P(\boldsymbol{y}|\boldsymbol{x}) = F_{T,N} \tag{2.6}$$

where $F_{T,N}$ may be computed through the following recursive relations:

$$F_{m,n} = \begin{cases} 0, & \text{if } m < 0 \text{ or } n < 0 \\ 1, & \text{if } m = n = 0 \\ P_t q(y_n|x_m) F_{m-1,n-1} + P_d F_{m-1,n} + \frac{P_i}{2} F_{m,n-1}, & \text{if } m < T \\ P_t q(y_n|x_m) F_{m-1,n-1} + P_d F_{m-1,n}, & \text{if } m = T \end{cases} \tag{2.7}$$

2.3 Drift Probability

Another quantity that later becomes instrumental is the drift probability. To compute this, we revisit the hidden Markov model illustrated in Fig. 2.1. Since the drift at any

given instant is just the difference between number of bits received and transmitted, the only distinction from the lattice metric described earlier, is that drift probability is not specific to the particular values of the received and transmitted symbols.

Lemma 2.3.1. *Consider a BSID channel where the insertion, deletion and substitution probabilities are P_i and P_d respectively. The probability distribution of drift d_T after the transmission of T bits over this channel, is given by:*

$$P(d_T = \delta) \quad = \quad \phi_{T,T+\delta} \tag{2.8}$$

where the RHS can be computed through the following recursive relations:

$$\phi_{m,n} = \begin{cases} 0, & \text{if } m < 0 \text{ or } n < 0 \\ 1, & \text{if } m = n = 0 \\ P_t\phi_{m-1,n-1} + P_d\phi_{m-1,n} + P_i\phi_{m,n-1}, & \text{if } m < T \\ P_t\phi_{m-1,n-1} + P_d\phi_{m-1,n}, & \text{else} \end{cases} \tag{2.9}$$

Proof. For simplicity, let N denote the number of bits received after the transmission of T bits. Since the channel starts in state S_0 and after transmission of T bits, ends in state S_T, we may write:

$$\begin{aligned} P(d_T = N - T) &= P(d_T = N - T, \{S_0, S_T\}) \\ &= \sum_{S \in \mathbb{S}} P(d_T = N - T, S) \end{aligned} \tag{2.10}$$

where $\mathbb{S}$ denotes the set of all possible paths from S_0 to S_T. Reconsidering the fact that final state S_T can only be reached from state S_{T-1} through transmission or deletion, we are led to the following recursive relation:

$$\begin{aligned} P(d_T = N - T, \{S_0, S_T\}) &= P_t P(d_{T-1} = (N-1) - (T-1), \{S_0, S_{T-1}\}) \\ &\quad + P_d P(d_{T-1} = N - (T-1), \{S_0, S_{T-1}\}) \\ &= P_t P(d_{T-1} = N - T, \{S_0, S_{T-1}\}) \\ &\quad + P_d P(d_{T-1} = N - T + 1, \{S_0, S_{T-1}\}) \end{aligned} \tag{2.11}$$

For further simplification, we need to obtain similar relations for intermediate states $S_{T-1}, S_{T-2}, \ldots$. To do so, we acknowledge that any such intermediate state can be entered either from its preceding state by a transmission or deletion, or from itself through a self-transition, which indicates an insertion. More explicitly, if the channel is in state S_m

where $m < T$, and n bits have been received:

$$\begin{aligned} P(d_m = n-m, \{S_0, S_m\}) &= P_t P(d_{m-1} = n-1-(m-1), \{S_0, S_{m-1}\}) \\ &\quad + P_d P(d_{m-1} = n-(m-1), \{S_0, S_{m-1}\}) \\ &\quad + P_i P(d_m = n-1-m, \{S_0, S_m\}) \end{aligned} \tag{2.12}$$

Denoting $P(d_m = n-m, \{S_0, S_m\})$ as $\phi_{m,n}$, (2.11) and (2.12) may be re-written as:

$$\phi_{m,n} = \begin{cases} P_t \phi_{m-1,n-1} + P_d \phi_{m-1,n} + P_i \phi_{m,n-1}, & \text{if } m < T \\ P_t \phi_{m-1,n-1} + P_d \phi_{m-1,n}, & \text{else} \end{cases} \tag{2.13}$$

Since prior to transmission, the channel always begins in state S_0 with the number of transmitted and received bits being initially 0, we also account for the following base cases:

$$\phi_{m,n} = \begin{cases} 0, & \text{if } m < 0 \text{ or } n < 0 \\ 1, & \text{if } m = n = 0 \end{cases} \tag{2.14}$$

Thus, to summarize if the number of transmitted and received bits are T and N respectively, the corresponding drift probability can be expressed as:

$$P(d_T = N - T) = \phi_{T,N} \tag{2.15}$$

where $\phi_{T,N}$ are computed according to (2.13) and (2.14). □

3 Sequential Decoding

The sequential decoding algorithm was first proposed by Wozencraft [**?** **?**] in 1957, and it remained the norm for decoding convolutional codes until the introduction of Viterbi algorithm [**?**] a decade later. The original method was subsequently refined into newer versions. Two of the most popular variants are the stack or ZJ algorithm [**?** **?**] and Fano's algorithm [**?**]. While the former is easier to understand and analyze, the latter significantly improves on the decoding efficiency.
Unlike a Viterbi decoder that works on the trellis diagram of a code, a sequential decoder uses the equivalent tree representation. It begins from the root and proceeds to further explore the tree by examining only a branch at a time, i.e. sequentially. In this manner, it conducts an efficient search operation to determine the most plausible path through this tree. The codeword indicated by the winning path provides the decoding estimate.
Since only a subset of all paths in the code tree are examined, a sequential decoder is essentially sub-optimal. However, on account of the fact that only one node is extended at any given time instant, decoding complexity is independent of encoder memory. This allows the use of codes with larger constraint lengths and therefore, sequential decoders can approach maximum-likelihood behavior asymptotically, with increasing encoder memory [**?**]. Additionally, the decoder's partial examination of the code tree may lead to a significant reduction in computational effort, especially in environments with low noise levels. It is however worth remembering that under conditions of higher noise, the decoder needs to backtrack more frequently and hence, may lose its computational advantage over Viterbi decoders.

3.1 Tree Codes

Sequential decoding was initially aimed at tree codes, which may be seen as a superset of convolutional codes. A tree encoder of rate $R = b/c$ essentially works on a predetermined tree-like structure, wherein 2^b branches emerge from each node corresponding to each possible input, and each branch is associated with a block of c symbols. Hence, each codeword corresponds to a particular sequence of branches in this code tree. A demonstrative example is stated below.

Example 3.1.1. *A tree code of rate* $R = 1/3$ *is illustrated in Fig. 3.1. As depicted, the branches directed upward correspond to input bit* 1*, and vice-versa. Let the provided information sequence be* 101*. The encoder begins from the root and traverses a path the*

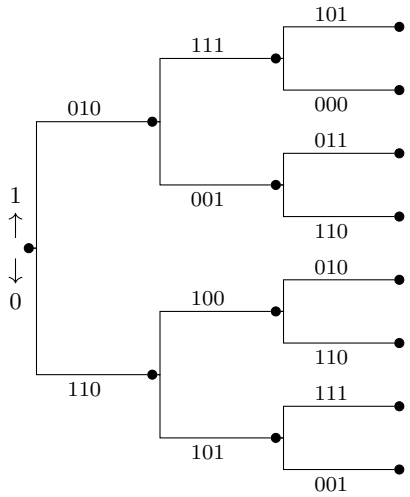


Figure 3.1: A binary tree code

code tree as dictated by the input bits. The final encoded frame is simply a concatenation of the edge labels along this path. Thus, the encoder output is 010 001 011*.*

3.2 Convolutional Codes

First introduced by Elias [**?**] in 1955, convolutional codes offer an alternative coding technique that can encode long message streams by treating them as a concatenation of smaller blocks. In contrast to block codes, they also incorporate memory. In particular, a convolutional code with the parameters (c, b, m) ensures that each output block of c bits not only depends on the current block of b information bits, but also on the preceding m input blocks. To implement a suitable encoder for the same, we may use a b-input, c-output linear sequential circuit with m memory cells. To exemplify this, we consider the following binary encoder.

Example 3.2.1. *The encoder for a binary* $(2, 1, 2)$ *convolutional code is illustrated in Fig. 3.2. We observe that this circuit includes* $m = 2$ *memory units, the contents of which indicate the state of this encoder. Thus, a total of four states are possible. It is also worth noting that the adders operate in* $\mathbb{F}_2$*.*
Now, consider a binary information stream $\boldsymbol{u} = (u_1, u_2, \cdots)$ *that is continuously fed into this circuit. At each time unit, the encoder receives a single bit, say* u_n*, and the two*

information bits that precede it, are shifted rightward into the memory cells. Thus, the two corresponding output bits produced, can be expressed as:

$$x_n^{(1)} = u_n + u_{n-1} + u_{n-2}$$
$$x_n^{(2)} = u_n + u_{n-2}$$

The above equations suggest that the output streams $\boldsymbol{x}^{(1)}$ *and* $\boldsymbol{x}^{(2)}$ *produced by the encoder, can be represented as results of the following convolution operations:*

$$\boldsymbol{x}^{(1)} = \boldsymbol{u} * \boldsymbol{g}^{(1)}$$
$$\boldsymbol{x}^{(2)} = \boldsymbol{u} * \boldsymbol{g}^{(2)}$$

where the vectors $\boldsymbol{g}^{(1)}$ *and* $\boldsymbol{g}^{(2)}$*, also known as generator polynomials, are given by:*

$$\boldsymbol{g}^{(1)} = (1 \quad 1 \quad 1)$$
$$\boldsymbol{g}^{(2)} = (1 \quad 0 \quad 1)$$

Consequently, the resulting encoded frame is:

$$\boldsymbol{x} = (x_1^{(1)}, x_2^{(1)}, x_2^{(1)}, x_2^{(2)}, \ldots)$$

By reinterpreting the convolutional encoder as a finite state machine, we can also equiv-

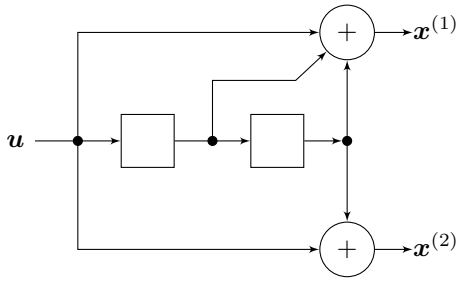


Figure 3.2: A $(2, 1, 2)$ binary convolutional encoder

alently represent it by means of a state diagram, as depicted in Fig. 3.3. Assuming that the initial state is always 00*, the final codeword* $\boldsymbol{x}$ *can also be obtained by traversing a path through this state diagram, as dictated by the information sequence* $\boldsymbol{u}$*.*

Since the state diagram of a convolutional encoder can also be transformed into a code tree representation as in Fig. 3.1, we can deduce that any convolutional code can be interpreted as a tree code. Hence, sequential decoders are equally applicable.

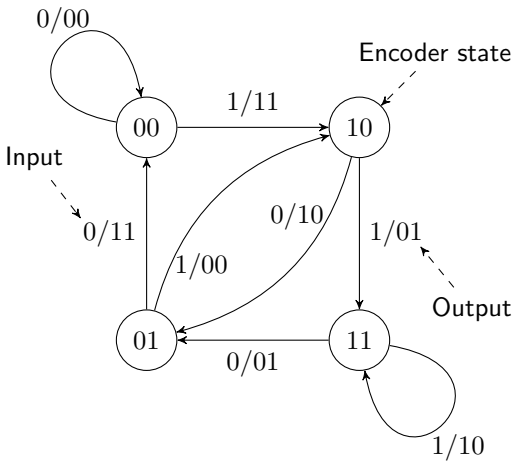


Figure 3.3: Encoder state diagram for a $(2, 1, 2)$ code

3.3 Transmission as Hidden Markov Model

In order to build a decoder for convolutional codes transmitted over a BSID channel, we choose to view the received sequence as the output of an HMM [**?** **?**], where a hidden state holds information on the encoder state and the drift induced by the channel. Thus, a hidden state at step t in this HMM, is essentially the pair (s_t, d_{ct}), where s_t is the encoder state, and d_{ct} denotes the net drift accumulated after the transmission of t output blocks, each of length c. Accordingly, the allowed state transitions would now involve both variables, as indicated in Fig. 3.4. Hence, to adapt the decoding process

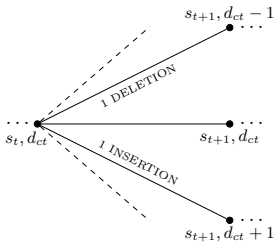


Figure 3.4: HMM seen by receiver: allowed transitions

of convolutional codes to the BSID channel, the associated code tree should now couple drift states with the existing encoder states. A demonstrative example is stated below.

Example 3.3.1. *Consider a* $(2, 1)$ *convolutional code, with generator polynomials being* $\boldsymbol{g}^{(1)} = (1 \quad 0)$ *and* $\boldsymbol{g}^{(2)} = (1 \quad 1)$. *The code has memory* 1 *and hence, two encoder states:*

s_0 *and* s_1.

Encoding starts with state s_0*, and initial drift is* 0*. The maximum number of insertions or deletions that can occur in a single frame is assumed to be* 1*. The resulting code tree is depicted in Fig. 3.5. Dashed and solid edges indicate inputs* 0 *and* 1 *respectively.*

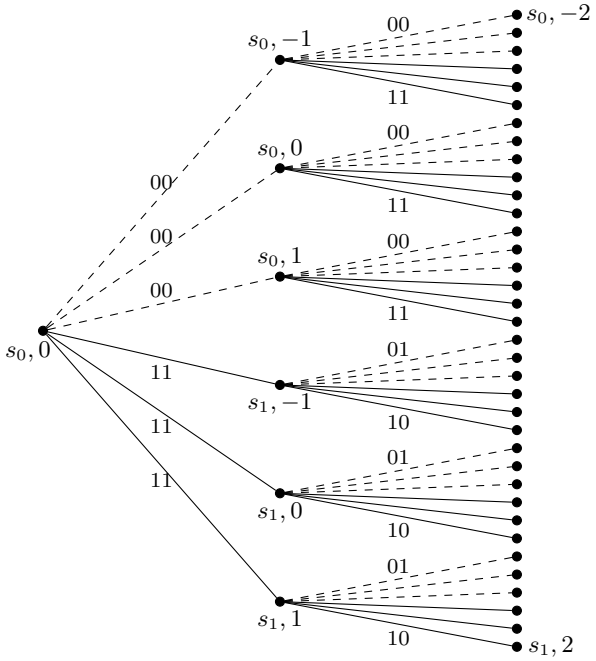


Figure 3.5: Code tree of a $(2, 1)$ convolutional code

3.3.1 Path merging for longer blocks

As an extension of Eg. 3.3.1, consider a $(4, 2)$ convolutional code specified by:

$$\begin{aligned} \boldsymbol{g}_1^{(1)} = \boldsymbol{g}_1^{(2)} = \boldsymbol{g}_1^{(4)} = \boldsymbol{g}_2^{(3)} = \boldsymbol{g}_2^{(4)} &= (1 \quad 0) \\ \boldsymbol{g}_1^{(3)} = \boldsymbol{g}_2^{(1)} &= (0 \quad 0) \\ \boldsymbol{g}_2^{(2)} &= (0 \quad 1) \end{aligned}$$

where $\boldsymbol{g}_i^{(j)}$ refers to the generator polynomial corresponding to the i^{th} input and j^{th} output. In this case, the encoding equations are given by:

$$\boldsymbol{x}^{(j)} = \boldsymbol{u}_1 * \boldsymbol{g}_1^{(j)} + \boldsymbol{u}_2 * \boldsymbol{g}_2^{(j)} \tag{3.1}$$

The first level of the associated code tree is depicted in Fig. 3.6. Here, a maximum of 2 insertions and deletions are considered per frame. On comparison with Fig. 3.5, we can verify that the respective codes are equivalent, i.e. both will encode a particular message vector into the same codeword. It is also worth noting that describing a message vector as a path in the $(4, 2)$ code tree, requires half of the number of branches needed in case of the $(2, 1)$ code tree.
One key difference however, is that a specific path in the $(4, 2)$ tree subsumes multiple paths in the $(2, 1)$ tree. For instance, consider the following path in the former:

$$(s_0, 0) \xrightarrow[(10)]{1101} (s_0, 0)$$

To compute its likelihood given a certain received vector, we may use (2.7). The resulting metric accounts for all possible state traversals of the channel, starting at S_0 and terminating at S_4 when transmission concludes. It is verifiable from the lattice diagram, that in each of these paths, the channel may exit state S_2 with a drift value varying from -2 to 2. Hence, the aforementioned branch includes the following paths in the $(2, 1)$ code tree:

$$\begin{aligned}
(s_0, 0) &\xrightarrow[(1)]{11} (s_0, -1) \xrightarrow[(0)]{01} (s_0, 0) \\
(s_0, 0) &\xrightarrow[(1)]{11} (s_0, 0) \xrightarrow[(0)]{01} (s_0, 0) \\
(s_0, 0) &\xrightarrow[(1)]{11} (s_0, 1) \xrightarrow[(0)]{01} (s_0, 0)
\end{aligned}$$

In addition, it also accounts for the following possibilities:

$$\begin{aligned}
(s_0, 0) &\xrightarrow[(1)]{11} (s_0, -2) \xrightarrow[(0)]{01} (s_0, 0) \\
(s_0, 0) &\xrightarrow[(1)]{11} (s_0, 2) \xrightarrow[(0)]{01} (s_0, 0)
\end{aligned}$$

Thus, despite the equivalence of these encoders, their code trees differ in behavior and an implicit merging of paths is observed in the code tree representing longer blocks. Naturally, this also leads to a fewer number of nodes in the tree for longer blocks, when considering message vectors of a fixed length.

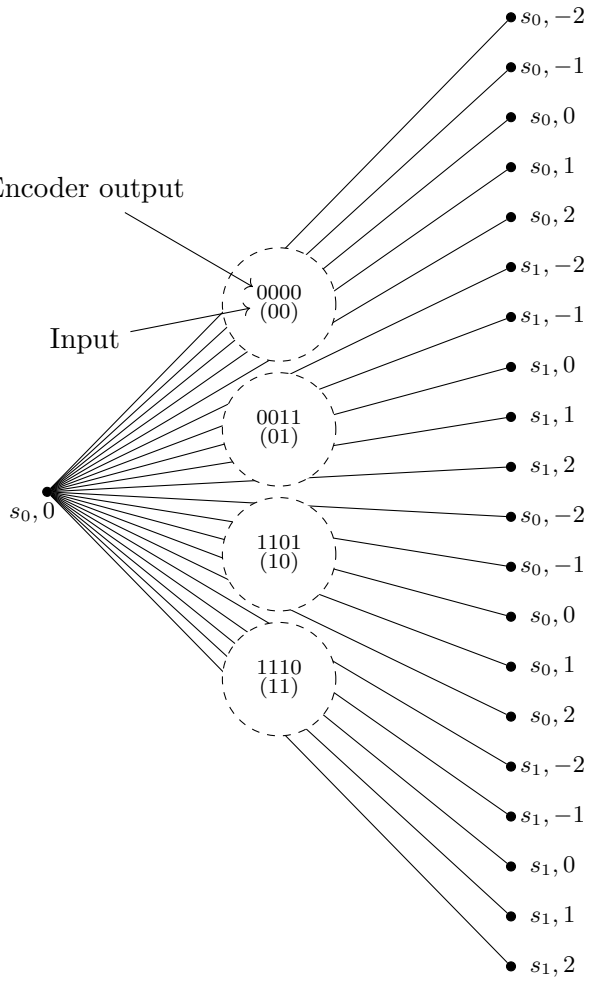


Figure 3.6: Code tree of a $(4, 2)$ convolutional code

3.4 Fano's Algorithm

In this work, we limit our focus to Fano's algorithm, which typically examines more paths than the stack algorithm and has lesser storage requirements. Like other sequential decoders, it tries to deduce from the received sequence, which path in the code tree was most likely undertaken. It is worth pointing out that in the context of a BSID channel, we are also interested in the sequence of drift values in addition to the encoder states. Thus the search operation is performed on an augmented code tree, similar to Fig. 3.5. Suppose that a bit sequence $\boldsymbol{y}$ was received due to the transmission of an encoded frame of B blocks. The decoder thus aims to find the most plausible path through the corresponding code tree of B levels. It begins from the root, examining the branches that immediately emanate from it. Each branch is associated with a metric value that quan-

tifies its likelihood with respect to $\boldsymbol{y}$. For a path $\boldsymbol{v}_l$ of length l, its total or cumulative metric denoted by $\mu(\boldsymbol{v}_l)$, is computed by adding up metrics of the l individual branches that constitute it. Hence, $\mu(\boldsymbol{v}_l)$ serves as a measure of closeness of the path $\boldsymbol{v}_l$ to $\boldsymbol{y}$. This metric is formally introduced and derived in the next section. Now since the goal is to find an encoded frame that bears the most resemblance with $\boldsymbol{y}$, it suffices to look for the path $\boldsymbol{v}_B^*$, that maximizes the cumulative metric, i.e.

$$\boldsymbol{v}_B^* = \arg\max_{\boldsymbol{v}_B} \mu(\boldsymbol{v}_B)$$

The codeword associated with $\boldsymbol{v}_B^*$ thus becomes the decoder's estimate.
The flowchart of Fano's algorithm is depicted in Fig. 3.7. Its operating principle is that along the correct path, branch metrics should, on an average, be positive. In other words, the cumulative metrics should follow a generally increasing trend. As we will see in succeeding chapters, this typically holds true. To exploit this feature, the algorithm makes use of a dynamic threshold, τ. This variable can only assume values that are integer multiples of the step size Δ, which is a user-defined parameter.
As mentioned previously, the decoder sequentially navigates the nodes in a tree, starting from the root and eventually concluding at a terminal node. At any given instant, the decoder can either move forward to one of the immediate successors, or back to the predecessor of the current node. Let μ_p, μ_c and μ_s denote the cumulative metrics of the predecessor, the current and the successor nodes respectively. The decoder tracks these values at each step, and is permitted to move forward only if μ_s is greater than or equal to threshold τ. Once the move is completed, the decoder's estimate is updated, and so are the variables μ_p and μ_c. Furthermore, if this node is being visited for the first time, threshold τ is tightened around its metric, μ_c as follows:

$$\tau \leq \mu_c < \tau + \Delta \tag{3.2}$$

If however, none of the successors uphold the threshold criterion, the decoder moves back and examines the next best successor. The decoder's estimate and metrics μ_p, μ_c and μ_c ar updated accordingly. In this manner, it systematically explores all the paths with metrics above or equal to the threshold τ.
To establish conditions that easily verify if a node is visited for the first time, we need to recognize that when the decoder backtracks to reach the node where threshold τ was tightened and increased to its present value, it lowers τ by the step size Δ, i.e. $\tau \leftarrow \tau - \Delta$. The decoder then moves forward again, repeating the branch extension routine with this lowered threshold. Thus, when this node is visited for a second time, $\tau + \Delta \leq \mu_c$ holds

true. This condition also holds for subsequent visits since τ can only dip lower, and consequently, can be used to detect first visits.
This mechanism of threshold lowering is crucial to prevent looping in the algorithm, as well as to ensure that eventually a terminal node is reached.

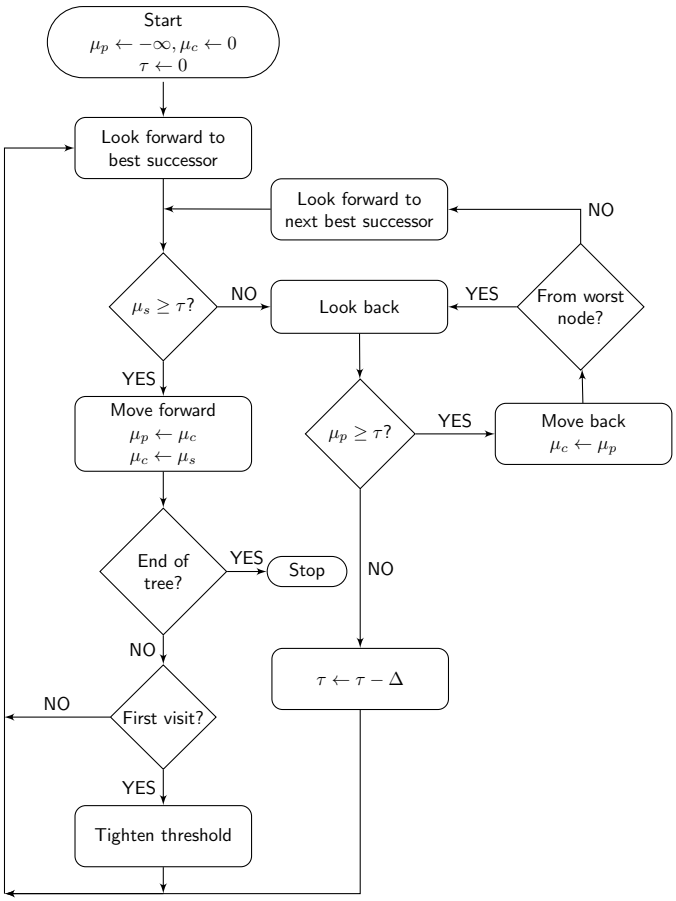


Figure 3.7: Flowchart of Fano's algorithm, [**?**]

From the preceding discussion, it seems evident that we need a metric to quantify the plausibility of a branch, and thus be able to rank successors. For this purpose, we derive a decoding metric for a BSID channel in the following section.

3.5 Decoding Metric

The aim of any path metric should be to appropriately quantify the closeness of a path's estimate to the sequence received. More specifically, it should help minimize the probability of an erroneous decision regarding the choice of successor node. To this end, we define the decoding metric of a path as the probability of its predicted codeword and drift

changes, given a specific received frame. This description is drawn from [**?**], wherein Massey proved the optimality of Fano metric, in the context of binary symmetric channels.

Let a received sequence, produced by the transmission of a codeword of B blocks, be $\boldsymbol{y} = \boldsymbol{y}_1^N = (y_1, \dots y_N)$. Now consider a path $\boldsymbol{v}_t$ in the corresponding code tree for a convolutional code (c, b), that hypothesizes the message vector $\boldsymbol{m}_1^{bt} = (m_1, \dots m_{bt})$ and drift state vector $\boldsymbol{d}_0^t = (d_0, d_c, \dots d_{ct})$, where initial drift $d_0 = 0$. Then the metric of node $\boldsymbol{v}_t$ in the tree, is given by:

$$\begin{aligned} \mu(\boldsymbol{v}_t) &= \log_2 P(\boldsymbol{v}_t|\boldsymbol{y}) \\ &= \log_2 P(\boldsymbol{v}_t, \boldsymbol{y}) - \log_2 P(\boldsymbol{y}) \\ &= \log_2 P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t, \boldsymbol{y}) - \log_2 P(\boldsymbol{y}) \end{aligned} \tag{3.3}$$

Before evaluating $P(\boldsymbol{v}_t, \boldsymbol{y})$, or the joint probability of path $\boldsymbol{v}_t$ and received frame $\boldsymbol{y}$, we need to recognize that $\boldsymbol{v}_t$ is not complete and only corresponds to the first t blocks of $\boldsymbol{y}$. To account for the remainder $\boldsymbol{y}_{ct+d_{ct}+1}^N$, we assume that it is produced by a tailing message sequence $\tilde{\boldsymbol{m}} = \tilde{\boldsymbol{m}}_1^{b(B-t)}$ of $B-t$ blocks. Additionally assuming that $\boldsymbol{y}_{ct+d_{ct}+1}^N$ is unaffected by bits transmitted prior to it, we may write:

$$\begin{aligned} P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t, \tilde{\boldsymbol{m}}, \boldsymbol{y}) &= P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t, \tilde{\boldsymbol{m}}) P(\boldsymbol{y}|\boldsymbol{m}, \boldsymbol{d}_0^t, \tilde{\boldsymbol{m}}) \\ &= P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) P(\tilde{\boldsymbol{m}}) P(\boldsymbol{y}_1^{ct+d_{ct}}|\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) P(\boldsymbol{y}_{ct+d_{ct}+1}^N|\tilde{\boldsymbol{m}}, d_t) \end{aligned}$$

Marginalizing this term over all possible message tails,

$$\begin{aligned} P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t, \boldsymbol{y}) &= \sum_{\tilde{\boldsymbol{m}} \in \mathbb{F}_2^{b(B-t)}} P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t, \tilde{\boldsymbol{m}}, \boldsymbol{y}) \\ &= P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) P(\boldsymbol{y}_1^{ct+d_{ct}}|\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) \sum_{\tilde{\boldsymbol{m}} \in \mathbb{F}_2^{b(B-t)}} P(\tilde{\boldsymbol{m}}) P(\boldsymbol{y}_{ct+d_{ct}+1}^N|\tilde{\boldsymbol{m}}, d_t) \end{aligned} \tag{3.4}$$

3.5.1 Probability of received vector

Equation (3.4) involves calculating the probability of receiving a bit sequence $\boldsymbol{y}_{ct+d_{ct}+1}^N$, due to the transmission of any frame of $B-t$ blocks. Similarly, evaluation of the decoding metric in (3.3) requires us to compute $P(\boldsymbol{y})$, which can be expressed as:

$$P(\boldsymbol{y}) = \sum_{\boldsymbol{m} \in \mathbb{F}_2^{bB}} P(\boldsymbol{y}, \boldsymbol{m}) \tag{3.5}$$

In either case, if the causal message vector is too long, such a marginalization operation becomes computationally impractical. To solve this, we attempt to simplify this probability calculation by assuming that all transmitted frames of a fixed length, are equally likely. For notational convenience, let the probability of receiving a vector $\boldsymbol{y}$ due to the transmission of a codeword containing T bits, be denoted by:

$$P_T(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \mathbb{F}_2^T} P(\boldsymbol{x}, \boldsymbol{y}) \tag{3.6}$$

Lemma 3.5.1. *Consider a BSID channel where the insertion, deletion and substitution probabilities are P_i, P_d and P_s respectively. Then, the probability of receiving a sequence $\boldsymbol{y}_1^N = (y_1, \cdots, y_N)$ given that T bits were transmitted, can be computed as:*

$$P_T(\boldsymbol{y}_1^N) \quad = \quad F'_{T,N} \tag{3.7}$$

where the RHS can be computed by means of the following recursive relations:

$$F'_{m,n} = \begin{cases} 0, & \text{if } m < 0 \text{ or } n < 0 \\ 1, & \text{if } m = n = 0 \\ \frac{P_t}{2} F'_{m-1,n-1} + P_d F'_{m-1,n} + \frac{P_i}{2} F'_{m,n-1}, & \text{if } m < T \\ \frac{P_t}{2} F'_{m-1,n-1} + P_d F'_{m-1,n}, & \text{else} \end{cases} \tag{3.8}$$

Proof. For the BSID channel, we note that the state transitions shown in Fig. 2.1 are specific to a given transmitted sequence. As before, let S_i denote the state reached by the channel after transmission of i symbols. To proceed with computing $P_T(\boldsymbol{y}_1^N)$, we attempt to remove the condition in (2.2), assuming that all T-bit codewords are equally likely.

$$\begin{aligned} P_T(\boldsymbol{y}_1^N) &= \sum_{\boldsymbol{x} \in \mathbb{F}_2^T} P(\boldsymbol{y}_1^N, \boldsymbol{x}) \\ &= \sum_{\boldsymbol{x} \in \mathbb{F}_2^T} P(\boldsymbol{x}) P(\boldsymbol{y}_1^N | \boldsymbol{x}) \\ &= \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T}} P(\boldsymbol{x}_1^T) P(\boldsymbol{y}_1^N, \{S_0, S_T\} | \boldsymbol{x}_1^T) \\ &= \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T}} P(\boldsymbol{x}_1^T) P_t q(y_N | x_T) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\} | \boldsymbol{x}_1^T) \\ &\quad + P_d \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T}} P(\boldsymbol{x}_1^T) P(\boldsymbol{y}_1^N, \{S_0, S_{T-1}\} | \boldsymbol{x}_1^T) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{x_T \in \mathbb{F}_2} P_t P(x_T) q(y_N|x_T) \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \\
&\quad + P_d \sum_{x_T \in \mathbb{F}_2} P(x_T) \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \\
&= P_t \sum_{x_T \in \mathbb{F}_2} P(x_T) q(y_N|x_T) \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \\
&\quad + P_d \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \qquad (3.9)
\end{aligned}
$$

The above terms are reducible using the following relation:

$$
\begin{aligned}
\sum_{x_T \in \mathbb{F}_2} P(x_T) q(y_N|x_T) &= P(x_T = 0) q(y_N|0) + P(x_T = 1) q(y_N|1) \\
&= \frac{1}{2}(1 - P_s) + \frac{1}{2} P_s = \frac{1}{2} \qquad (3.10)
\end{aligned}
$$

Since $\sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T)$ is unaffected by x_T, the above relation can be incorporated in (3.9) as follows:

$$
\begin{aligned}
P_T(\boldsymbol{y}_1^N) &= \frac{P_t}{2} \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N-1}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \\
&\quad + P_d \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le T-1}} P(\boldsymbol{x}_1^{T-1}) P(\boldsymbol{y}_1^{N}, \{S_0, S_{T-1}\}|\boldsymbol{x}_1^T) \qquad (3.11)
\end{aligned}
$$

As in case of the likelihood metric, we again observe that in order to obtain a recurrence relation for $\sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le m}} P(\boldsymbol{x}_1^m) P(\boldsymbol{y}_1^n, \{S_0, S_m\}|\boldsymbol{x}_1^T)$, where S_m is any intermediate state, we need to additionally account for the insertion edge from state S_m:

$$
\begin{aligned}
\sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le m}} P(\boldsymbol{x}_1^m) P(\boldsymbol{y}_1^n, \{S_0, S_m\}|\boldsymbol{x}_1^T) &= \frac{P_t}{2} \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le m-1}} P(\boldsymbol{x}_1^{m-1}) P(\boldsymbol{y}_1^{n-1}, \{S_0, S_{m-1}\}|\boldsymbol{x}_1^T) \\
&\quad + P_d \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le m-1}} P(\boldsymbol{x}_1^{m-1}) P(\boldsymbol{y}_1^{n}, \{S_0, S_{m-1}\}|\boldsymbol{x}_1^T) \\
&\quad + \frac{P_i}{2} \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \le i \le m}} P(\boldsymbol{x}_1^{m}) P(\boldsymbol{y}_1^{n-1}, \{S_0, S_m\}|\boldsymbol{y}_1^T) \qquad (3.12)
\end{aligned}
$$

The recursive relations described in (3.11) and (3.12) suggest that to compute $P_T(\boldsymbol{y}_1^N)$,

we may reuse the lattice structure defined in the previous chapter. To simplify notations, let

$$F'_{m,n} = \sum_{\substack{x_i \in \mathbb{F}_2 \\ 1 \leq i \leq m}} P(\boldsymbol{x}_1^m) P(\boldsymbol{y}_1^n, \{S_0, S_m\} | \boldsymbol{x}_1^T)$$

for a specific received sequence $\boldsymbol{y}_1^N$. Hence (3.11) and (3.12) may be restated:

$$F'_{m,n} = \begin{cases} \frac{P_t}{2} F'_{m-1,n-1} + P_d F'_{m-1,n} + \frac{P_i}{2} F'_{m,n-1}, & \text{if } 0 \leq m < T \\ \frac{P_t}{2} F'_{m-1,n-1} + P_d F'_{m-1,n}, & \text{if } m = T \end{cases} \tag{3.13}$$

As before, the base cases outlined in (2.5) also hold here. The only distinction from the computation of likelihood metric is due to weight of the diagonal edge, which is now the constant, $\frac{P_t}{2}$. Finally, $P_T(\boldsymbol{y}_1^N)$ is given by:

$$P_T(\boldsymbol{y}_1^N) = F'_{T,N} \tag{3.14}$$

We also observe that the final expression only depends on the number of received and transmitted bits. This implies that, for a fixed number of transmitted bits, all received sequences of a specific length are equally probable. □

Using the above results, (3.3) and (3.4) may be combined as follows:

$$\mu(\boldsymbol{v}_t) \quad = \quad \log_2 P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) P(\boldsymbol{y}_1^{ct+d_{ct}} | \boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) + \log_2 \frac{P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^N)}{P_{cB}(\boldsymbol{y}_1^N)} \tag{3.15}$$

Since the message $\boldsymbol{m}$ is independent of the drift errors induced by the channel, it is easy to see that:

$$P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) = P(\boldsymbol{m}_1^{bt}) P(\boldsymbol{d}_0^t) \tag{3.16}$$

Moreover, since the individual b-bit information blocks in $\boldsymbol{m}_1^{bt}$ are independent of each other, $P(\boldsymbol{m}_1^{bt})$ can be simplified as:

$$P(\boldsymbol{m}_1^{bt}) = \prod_{i=1}^{t} P(\boldsymbol{m}_{b(i-1)+1}^{bi}) \tag{3.17}$$

Normally, it can be reasonably assumed that all b-bit message blocks are equiprobable. However, if a certain message block $\boldsymbol{m}_{b(i-1)+1}^{bi}$ belongs to the terminating zero tail, we

may write that $P(\boldsymbol{m}_{b(i-1)+1}^{bi} = \mathbf{0}) = 1$.

$$P(\boldsymbol{m}_{b(i-1)+1}^{bi}) = \begin{cases} 1, & \text{if } \boldsymbol{m}_{b(i-1)+1}^{bi} = \mathbf{0} \text{ lies in terminating zero tail} \\ 0, & \text{if } \boldsymbol{m}_{b(i-1)+1}^{bi} \neq \mathbf{0} \text{ lies in terminating zero tail} \\ 2^{-b}, & \text{else} \end{cases} \tag{3.18}$$

Now in regard to the probability of a drift sequence, we can infer from Fig. 2.1 that any sequence of drift states behaves like a Markov chain. This observation simplifies the task of evaluating $P(\boldsymbol{d})$ as follows:

$$\begin{aligned} P(\boldsymbol{d}_0^t) &= P(d_0) \prod_{i=1}^{t} P(d_{ci}|d_{c(i-1)}) \\ &= \prod_{i=1}^{t} P(d_{ci}|d_{c(i-1)}) \end{aligned} \tag{3.19}$$

The second equality follows from the fact that the initial drift is always set to $d_0 = 0$. Also, we only evaluate $P(\boldsymbol{d}_0^t)$ for drift sequences with $d_0 = 0$. Using (3.17) and (3.19), we can simplify (3.16) as:

$$P(\boldsymbol{m}_1^{bt}, \boldsymbol{d}) = \prod_{i=1}^{t} P(\boldsymbol{m}_{b(i-1)+1}^{bi}) P(d_{ci}|d_{c(i-1)}) \tag{3.20}$$

In a similar fashion, the term $P(\boldsymbol{y}_1^{ct+d_{ct}}|\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t)$ in (3.15) can also be decomposed. Let the transmitted codeword corresponding to message $\boldsymbol{m}_1^{bt}$ be denoted by $\boldsymbol{x}_{\boldsymbol{m}}$. Given $\boldsymbol{x}_{\boldsymbol{m}}$ and the drift states $\boldsymbol{d}_0^t$, we can treat the individual blocks in $\boldsymbol{y}_1^{ct+d_{ct}}$ independently. Hence,

$$\begin{aligned} P(\boldsymbol{y}_1^{ct+d_{ct}}|\boldsymbol{m}_1^{bt}, \boldsymbol{d}_0^t) &= P(\boldsymbol{y}_1^{ct+d_{ct}}|\boldsymbol{x}_{\boldsymbol{m}}, \boldsymbol{d}_0^t) \\ &= \prod_{i=0}^{t-1} P(\boldsymbol{y}_{ci+d_{ci}+1}^{c(i+1)+d_{c(i+1)}}|\boldsymbol{x}_{ci+1}^{c(i+1)}, d_{ci}, d_{c(i+1)}) \end{aligned} \tag{3.21}$$

Incorporating (3.20) and (3.21) in (3.15), we get:

$$\begin{aligned} \mu(\boldsymbol{v}_t) &= \log_2 \prod_{i=0}^{t-1} P(\boldsymbol{m}_{bi+1}^{b(i+1)}) + \log_2 \prod_{i=0}^{t-1} P(d_{c(i+1)}|d_{ci}) \\ &\quad + \log_2 \prod_{i=0}^{t-1} P(\boldsymbol{y}_{ci+d_{ci}+1}^{c(i+1)+d_{c(i+1)}}|\boldsymbol{x}_{ci+1}^{c(i+1)}, d_{ci}, d_{c(i+1)}) + \log_2 \frac{P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^{N})}{P_{cB}(\boldsymbol{y})} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{t-1} \log_2 \left(P(\boldsymbol{m}_{bi+1}^{b(i+1)}) P(d_{c(i+1)}|d_{ci}) P(\boldsymbol{y}_{ci+d_{ci}+1}^{c(i+1)+d_{c(i+1)}} | \boldsymbol{x}_{ci+1}^{(i+1)c}, d_{ci}, d_{c(i+1)}) \right) \\
&\quad + \log_2 \frac{P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^{N})}{P_{cB}(\boldsymbol{y})} \\
&= \sum_{i=0}^{t-1} \log_2 \left(P(\boldsymbol{m}_{bi+1}^{b(i+1)}) P(\boldsymbol{y}_{ci+d_{ci}+1}^{c(i+1)+d_{c(i+1)}}, d_{c(i+1)} | \boldsymbol{x}_{ci+1}^{c(i+1)}, d_{ci}) \right) \\
&\quad + \log_2 P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^{N}) - \log_2 P_{cB}(\boldsymbol{y}) \qquad (3.22)
\end{aligned}
$$

where branch likelihood $P(\boldsymbol{y}_{ci+d_{ci}+1}^{c(i+1)+d_{c(i+1)}}, d_{c(i+1)} | \boldsymbol{x}_{ci+1}^{c(i+1)}, d_{ci})$ may be calculated using (2.7). Instead of re-computing all branch likelihoods from the root to compute the metric of a node, we may equivalently define a branch metric as follows:

$$
\begin{aligned}
Z(\boldsymbol{v}_{t-1} \to \boldsymbol{v}_t) &= \mu(\boldsymbol{v}_t) - \mu(\boldsymbol{v}_{t-1}) \\
&= \log_2 P(\boldsymbol{m}_{b(t-1)+1}^{bt}) + \log_2 P(\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{ct+d_{ct}}, d_{ct} | \boldsymbol{x}_{c(t-1)+1}^{ct}, d_{c(t-1)}) \\
&\quad + \log_2 \frac{P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^{N})}{P_{c(B-t+1)}(\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{N})} \qquad (3.23)
\end{aligned}
$$

Since the decoder continuously tracks the metric of its current node, this branch metric can be easily used as an update term to evaluate the node metric of a chosen successor.

Example 3.5.1. *Consider the* $(2, 1, 1)$ *convolutional code from Eg. 3.3.1. Assume that a zero-terminated codeword containing 2 information blocks is transmitted over a BSID channel with the insertion, deletion and substitution probabilities being* 0.002*,* 0.003 *and* 0.01 *respectively. The following sequence is received:*

$$\boldsymbol{y} = (1\ 1\ 0\ 1\ 0\ 0\ 1)$$

Since the final drift is 1 *and the encoder ends at state* s_0*, we only consider paths in the code tree that terminate at* $(s_0, 1)$*. We assume that a maximum of* 1 *insertion or deletion can occur in a single block. The node metrics are computed with (3.22) and shown in Fig. 3.8. The step size is set to* $\Delta = 0.5$*.*
In the following, we use the following abbreviations to indicate the decoder's decision in reference to the algorithm's flowchart in Fig. 3.7:

- *MF: Move forward*
- *MB: Move back*
- *NB: Look forward to next best successor*

- *LT: Lower threshold*

Fano's decoder undertakes the following steps.

Iter.	$\boldsymbol{v_c}$	μ_p	μ_c	μ_s	τ	*Action*
0	$(s_0, 0)$	$-\infty$	0	0.39	0	*MF*
1	$(s_0, 0) \to (s_1, 0)$	0	0.39	0.36	0	*MF*
2	$(s_0, 0) \to (s_1, 0) \to (s_0, 0)$	0.39	0.36	-4.3	0	*MB, NB*
3	$(s_0, 0) \to (s_1, 0)$	0	0.39	-0.63	0	*MB, NB*
4	$(s_0, 0)$	$-\infty$	0	-7.21	0	*LT*
5	$(s_0, 0)$	$-\infty$	0	0.39	-0.5	*MF*
6	$(s_0, 0) \to (s_1, 0)$	0	0.39	0.36	-0.5	*MF*
7	$(s_0, 0) \to (s_1, 0) \to (s_0, 0)$	0.39	0.36	-4.3	-0.5	*MB, NB*
8	$(s_0, 0) \to (s_1, 0)$	0	0.39	-0.63	-0.5	*MB, NB*
9	$(s_0, 0)$	$-\infty$	0	-7.21	-0.5	*LT*
10	$(s_0, 0)$	$-\infty$	0	0.39	-1	*MF*
11	$(s_0, 0) \to (s_1, 0)$	0	0.39	0.36	-1	*MF*
12	$(s_0, 0) \to (s_1, 0) \to (s_0, 0)$	0.39	0.36	-4.3	-1	*MB, NB*
13	$(s_0, 0) \to (s_1, 0)$	0	0.39	-0.63	-1	*MF*
14	$(s_0, 0) \to (s_1, 0) \to (s_1, 1)$	0.39	-0.63	1.34	-1	*MF*
15	$(s_0, 0) \to (s_1, 0) \to (s_1, 1) \to (s_0, 1)$	-0.63	1.34		-1	

The decoder thus chooses the path

$$(s_0, 0) \to (s_1, 0) \to (s_1, 1) \to (s_0, 1)$$

and the estimated codeword is:

$$\boldsymbol{x}^* = (1\ 1\ \ 1\ 0\ \ 0\ 1)$$

From the above example, it is easy to see how errors in transmission lead to a reduction in node metric, thus forcing the decoder to explore other alternatives. In this manner, higher noise can increase the number of computations. This problem can be mitigated to some extent by choosing the step size appropriately. For instance, if step size was set to $\Delta = 1$ in the preceding example, the decoder would need to track back to the root just once, leading to fewer iterations. However, choosing an extremely high value will reduce the threshold excessively and thus lead to the selection of incorrect branches. The choice of step size hence play a crucial role in determining the speed of Fano's decoder.

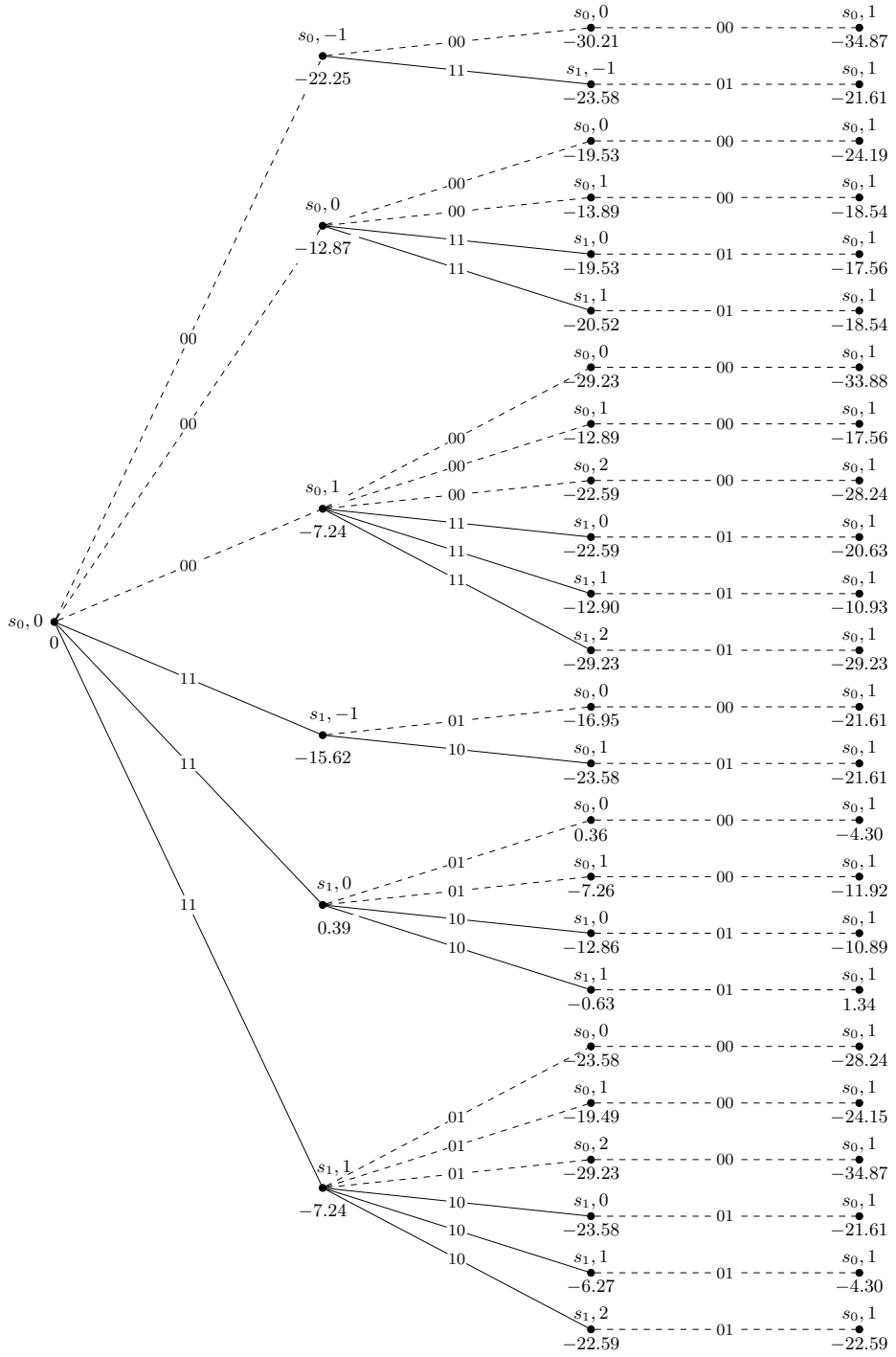


Figure 3.8: Code tree of $(2, 1)$ convolutional code

4 Computational Analysis of Fano's Algorithm

Due to the very nature of Fano's algorithm, the number of computations required to decode any received frame is essentially a random variable. Evidently, this is in direct contrast with the Viterbi algorithm which always involves the same amount of decoding effort for a given code, regardless of the received frame and its extent of corruption. Past attempts at analyzing the average decoding complexity of sequential decoders primarily rely on either conducting an asymptotic analysis over a random ensemble of codes [**?** **?** **?**], or modeling the distribution of branch metrics across the code tree as a multi-type branching process [**?** **?** **?**]. The approach used here is based on the former strategy and is specifically adopted from the analysis outlined in [**?**].
Referring back to Fig. 3.7, we observe that the most computationally demanding step of Fano's algorithm involves looking forward to all successors of the current node and ranking them, in an attempt to further extend the current path. This ranking is obtained by computing the respective branch metrics of all immediate successors using (3.23). Now, since the decoder executes this step each time a new node is visited, we can characterize the time complexity of a single decoding operation as a product of the total number of visits to all nodes in the code tree, and the time complexity of a single ranking operation.

4.1 Ranking Complexity

Given a convolutional code (c, b), the number of successors of a any given node in the code tree, denoted henceforth as $\mathcal{D}$, is given by:

$$\mathcal{D} = 2^b(i_{\max} + d_{\max} + 1) \tag{4.1}$$

where $i_{\max}$ and $d_{\max}$ refer to the maximum number of insertions and deletions considered per block. Hence, during every ranking operation the decoder is required to compute the respective branch metrics for each of these $\mathcal{D}$ successors with (3.23), which is restated as

follows:

$$\begin{aligned} Z(\boldsymbol{v}_{t-1} \to \boldsymbol{v}_t) &= \mu(\boldsymbol{v}_t) - \mu(\boldsymbol{v}_{t-1}) \\ &= \log_2 P(\boldsymbol{m}_{b(t-1)+1}^{bt}) + \log_2 P(\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{ct+d_{ct}}, d_{ct} | \boldsymbol{x}_{c(t-1)+1}^{ct}, d_{c(t-1)}) \\ &\quad + \log_2 \frac{P_{c(B-t)}(\boldsymbol{y}_{ct+d_{ct}+1}^{N})}{P_{c(B-t+1)}(\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{N})} \end{aligned} \tag{4.2}$$

In regard to the last term involving a ratio of tail probabilities, we may recall from (3.7) that for a given number of transmitted bits, all received sequences of a fixed length are equally probable. Thus, these tail probability values may be pre-stored, in which case the task of computing the decoding metric of a branch is reduced to simply calculating the logarithmic branch likelihood, i.e. $\log_2 P(\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{ct+d_{ct}}, d_{ct} | \boldsymbol{x}_{c(t-1)+1}^{ct}, d_{c(t-1)})$. As seen in Section 2.2, the likelihood of receiving a bit sequence $\boldsymbol{y} = (y_1, \cdots, y_N)$, given that the transmitted vector was $\boldsymbol{x} = (x_1, \cdots, x_T)$, is computed with a lattice structure, similar to that depicted in Fig. 2.2. Evidently, the complexity of computing this metric is dictated by the size of this lattice, which is given by $(T+1)(N+1)$. Thus, the complexity of computing the branch metrics for each of the $\mathcal{D}$ successors, is proportional to:

$$2^b \sum_{d=-d_{\max}}^{i_{\max}} (c+1)(c+d+1)$$

An interesting aspect of using the lattice structure, is that once $P(\boldsymbol{y}|\boldsymbol{x})$ has been evaluated, the values of $P(\boldsymbol{y}_1^{N-1}|\boldsymbol{x})$, $P(\boldsymbol{y}_1^{N-2}|\boldsymbol{x})$ etc. become readily available from elements of the last row in the lattice. As a result, repeating these metric computations for each successor is redundant and it suffices to simply calculate the branch metric that allows the most insertions, for each of the 2^b possible inputs. This reduces the complexity of the ranking operation to:

$$2^b(c+1)(c+i_{\max}+1) \tag{4.3}$$

Quite notably, this term is independent of encoder memory and thus helps illustrate why the complexity of sequential decoders remains unaffected by longer constraint lengths.

4.2 Analysis

As stated earlier, the most costly step in Fano's algorithm is the evaluation of a node's successors by means of lattice metric calculations. Thus, in order to determine the average complexity of Fano's decoder, we aim to simply count the average number of times a decoder moves forward to a new node and examines the branches that emerge from

it. For the sake of simplicity, we henceforth refer to the process of evaluating a node's immediate successors, as a visit to this node.

The aim of this section is to establish an upper bound on the total number of visits to the nodes in a given code tree, as it is quite representative of the total number of computations performed during decoding. For a given received frame, this quantity is essentially a random variable that depends on the associated message sequence, channel noise and lastly, the code itself.

We begin by studying the decoding complexity for a specific frame. Consider a received vector $\boldsymbol{y} = \boldsymbol{y}_1^N = (y_1, \ldots, y_N)$ which results from the transmission of L blocks, and let the decoding result correspond to the following path:

$$p^* = ((s_0^*, d_0^*), (s_1^*, d_1^*), \ldots (s_L^*, d_L^*))$$

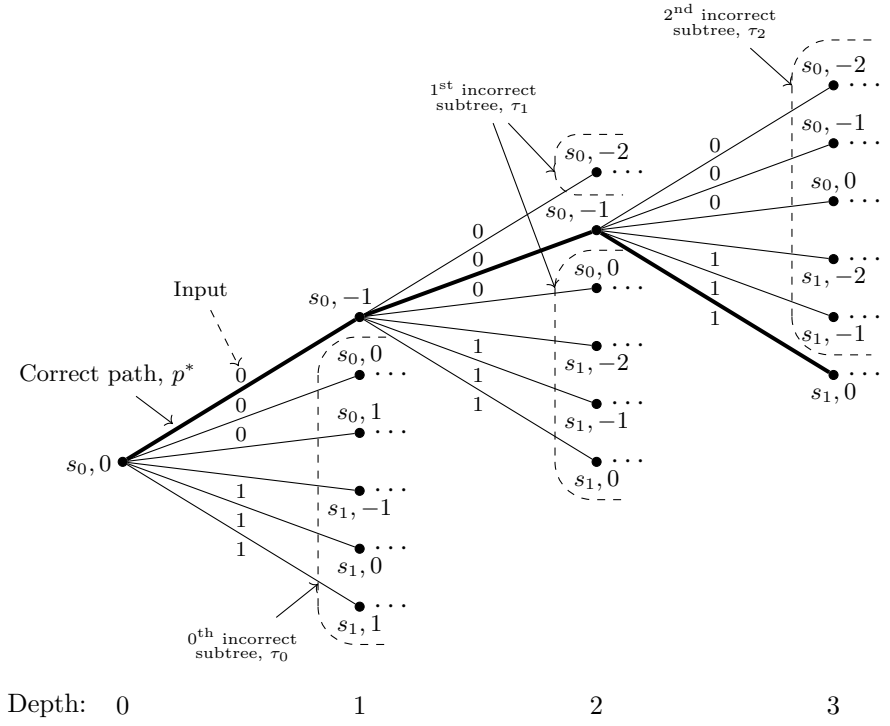


Figure 4.1: Incorrect subtrees in a code tree.

The associated code tree is illustrated in Fig. 4.1. Assuming that p^* predicts the transmitted codeword accurately, the overall decoding effort for this particular frame effectively consists of the computations performed on nodes that either lie on p^*, or an incorrect path. To simplify the matter, we group nodes of the latter category according to the in-

correct subtree in which they lie. In this context, an incorrect subtree is defined as follows. Let $\boldsymbol{v}_j = (\boldsymbol{s}_0^j, \boldsymbol{d}_0^j)$ denote a node in the code tree that describes the path $((s_0, d_0), \ldots, (s_j, d_j))$. Also, let the i^{th} incorrect subtree be denoted by τ_i. As depicted in Fig. 4.1, τ_i is the set of all nodes that hypothesize any false path that stems from $\boldsymbol{v}_i^*$, i.e. i^{th} node on the correct path, p^*. Thus the total number of computations required to decode $\boldsymbol{y}$, can be expressed as:

$$\sum_{i=0}^{L-1} \left(C'(\boldsymbol{v}_i^*) + \sum_{j=i+1}^{L+1} \sum_{\boldsymbol{v}_j \in \tau_i} C'(\boldsymbol{v}_j) \right) \tag{4.4}$$

where $C'(\boldsymbol{v}_j)$ denotes the number of times node $\boldsymbol{v}_j$ is visited.
The quantity $C'(\boldsymbol{v}_i^*) + \sum_{j=i+1}^{L+1} \sum_{\boldsymbol{v}_j \in \tau_i} C'(\boldsymbol{v}_j)$ simply denotes the total number of visits to $\boldsymbol{v}_i^*$ and any node in τ_i. We may interpret this as the number of computations required to correctly decode the i^{th} block. Denoting this by $C(\boldsymbol{v}_i^*)$, we conclude that decoding $\boldsymbol{y}$ requires a total of $\sum_{i=0}^{L-1} C(\boldsymbol{v}_i^*)$ computations.
By averaging $C(\boldsymbol{v}_i^*)$ over a code ensemble and all possible transmitted and transmitted sequences, we arrive at the mean number of computations for correct decoding of i^{ith} block. For simplicity, we consider an infinite-memory convolutional code (c, b, ∞), since this allows us to assume similar statistical properties of all incorrect subtrees, which now have infinite depth. This in turn implies that all $C(\boldsymbol{v}_i^*)$'s, where $i = 0, 1, \ldots$, have the same distribution. Consequently, the task of analyzing the complexity of Fano's algorithm can be reduced to determining the average number of computations for correctly decoding a single block. Denoting this by C_{av}, we may write:

$$C_{\text{av}} = E[C(\boldsymbol{v}_i^*)] = E[C(\boldsymbol{v}_0^*)]$$

Since it suffices to obtain an upper bound for $E[C(\boldsymbol{v}_0^*)]$, we simply need to determine the number of times root node $\boldsymbol{v}_0^*$, and additionally any node in the corresponding incorrect subtree, τ_0, are visited. To do so, we need to recall three key features of Fano's algorithm:

P1 A particular node $\boldsymbol{v}_j$ may be visited only if its metric $\mu(\boldsymbol{v}_j)$ lies above the current threshold T.

P2 For a particular value of T, any node may be visited at most once.

P3 When a node $\boldsymbol{v}_j$ is visited for the first time, threshold T is tightened about its metric as follows:

$$T \leq \mu(\boldsymbol{v}_j) < T + \Delta$$

For each subsequent visit, the threshold value is lowered by Δ.

While properties P1 and P3 follow directly from the algorithm itself, P2 may be inferred from the fact that when the decoder cannot find any path that stays above the current threshold, it moves back to the node where this threshold value was set, decrements it by the step size and then resumes its search for the best path. This naturally ensures that no node is revisited with the same threshold.
Now, let the minimum metric along the correct path p^* be defined as:

$$\mu^*_{\min} = \min_{j \geq 0} \mu(\boldsymbol{v}^*_j) \tag{4.5}$$

Also let $T^*_{\min}$ denote the minimum value assumed by the threshold during decoding. From property P1, it is easy to see that $\mu^*_{\min} \geq T^*_{\min}$. Furthermore, since property P2 implies that consecutive threshold values are spaced apart by Δ, we can infer that:

$$\mu^*_{\min} - \Delta < T^*_{\min} \leq \mu^*_{\min} \tag{4.6}$$

As demonstrated previously in Eg. 3.5.1, a reduction in metric along the correct path prompts the examination of alternate paths in the incorrect subtrees. Additionally from property P1, we can deduce that lower the value of $T_{\min}$, more is the number of incorrect nodes visited. Hence, the net decoding effort is heavily dependent on $T^*_{\min}$. Consequently, we may simplify the computation of C_{av} as follows:

$$\begin{aligned} C_{\text{av}} &= E[C(\boldsymbol{v}^*_0)] \\ &= \sum_{y \in \mathbb{T}} P(T^*_{\min} = y) E[C(\boldsymbol{v}^*_0) | T^*_{\min} = y] \end{aligned} \tag{4.7}$$

where $\mathbb{T}$ denotes the set of all values that $T^*_{\min}$ may assume. Since decoding begins at root with an initial threshold $T = 0$, which can subsequently be incremented or decremented by Δ, possible values of $T^*_{\min}$ comprise $0, -\Delta, -2\Delta$ and so on. Thus,

$$C_{\text{av}} = \sum_{i=0}^{\infty} P(T^*_{\min} = -i\Delta) E[C(\boldsymbol{v}^*_0) | T^*_{\min} = -i\Delta] \tag{4.8}$$

4.2.1 Metric behavior along correct path

From (4.6), it is easy to see that in order to determine the probability distribution of the minimum threshold, we need to in turn investigate the distribution of Fano branch metrics along the correct path, i.e. the path in a code tree that most closely describes the transmitted codeword as well as the drift events that transpired.
For a specific received frame $\boldsymbol{y}$, let $p^* = (\boldsymbol{v}^*_0, \boldsymbol{v}^*_1, \ldots)$ denote the correct path in a code

tree, with the node metrics along the levels being $\mu_0^*, \mu_1^*, \mu_2^*, \ldots$, where $\mu_0^* = 0$. The node metric at depth i for $i \geq 1$, is expressible as:

$$\mu_i^* = \sum_{j=0}^{i-1} Z_j^* \tag{4.9}$$

where $Z_j^* = Z(\boldsymbol{v}_j^* \rightarrow \boldsymbol{v}_{j+1}^*)$ refers to the metric of the $(j+1)^{\text{th}}$ branch along p^*. Due to the memory feature of convolutional codes, we know that any block in a given codeword is influenced by those preceding it. As a consequence, the branch metrics along a correct path are similarly related. However for analytical convenience, we assume that Z_j^*'s are independent and identically distributed. This allows us to write that:

$$\begin{aligned} E[2^{\sigma\mu_i^*}] &= E[2^{\sigma \sum_{j=0}^{i-1} Z_j^*}] \\ &= (E[2^{\sigma Z^*}])^i \\ &= g_0(\sigma)^i \end{aligned} \tag{4.10}$$

where $g_0(\sigma)$ refers to the moment generating function of the branch metric along the correct path. To proceed with computing $P(\mu_j^* < y)$, we note that for any $\sigma < 0$,

$$P(\mu_j^* < y) = P(2^{\sigma\mu_j^*} > 2^{\sigma y})$$

Since $2^{\sigma\mu_j^*}$ is always non-negative, we may apply Markov's inequality:

$$\begin{aligned} P(\mu_j^* < y) &= P(2^{\sigma\mu_j^*} > 2^{\sigma y}) \\ &\leq 2^{-\sigma y} E[2^{\sigma\mu_i^*}] \\ &= 2^{-\sigma y} g_0(\sigma)^i \end{aligned}$$

Choosing a suitable $\sigma_0 < 0$ such that $g_0(\sigma_0) = 1$, the above inequality becomes:

$$P(\mu_i^* < y) \leq 2^{-\sigma_0 y} \tag{4.11}$$

As this upper bound does not depend on depth of node i, we may write that:

$$P(\mu_{\min}^* < y) \leq 2^{-\sigma_0 y} \tag{4.12}$$

Now, since it has been established that $T_{\min}^*$ can either be 0 or a negative multiple of Δ, its cumulative distribution function can be expressed as:

$$P(T_{\min}^* \leq y) = P(T_{\min}^* = y') + P(T_{\min}^* = y' - \Delta) + P(T_{\min}^* = y' - 2\Delta) + \ldots$$

where y' is the largest multiple of Δ, such that $y' \leq y$.
Using property P3 and (4.12), we can reduce this relation to:

$$\begin{aligned} P(T^*_{\min} \leq y) &= P(y' \leq \mu^*_{\min} < y' + \Delta) + P(y' - \Delta \leq \mu^*_{\min} < y') \\ &\quad + P(y' - 2\Delta \leq \mu^*_{\min} < y' - \Delta) + \dots \\ &= P(\mu^*_{\min} < y' + \Delta) \\ &\leq 2^{-\sigma_0(y'+\Delta)} \\ &\leq 2^{-\sigma_0(y+\Delta)} \end{aligned} \tag{4.13}$$

4.2.2 Asymptotic approximation of branch metric

To characterize the distribution of minimum threshold in the preceding section, we assumed that the branch metrics along the correct path were independent and identically distributed. This assumption certainly holds for the binary symmetric channel [**?**]. In case of the BSID channel however, the assumption of identical distribution certainly does not hold, on account of the message likelihood term that vanishes towards the tail section, and the ratio of tail probabilities, as seen in (4.2).
We can circumvent this issue for the purpose of analysis, by obtaining an asymptotic approximation which would be valid here since we are considering an ensemble of codes with infinite memory. To proceed along these lines, we first attempt to approximate the ratio of tail probabilities in (4.2). This can be accomplished using the theory on lattice walks and analytic combinatorics.
Referring back to (3.7) which helps compute the probability of receiving a frame of N bits given that T bits were transmitted, we recognize that $P_T(\boldsymbol{y}_1^N)$ can be interpreted as a weighted sum of paths through a two-dimensional lattice. These paths begin at origin $(0, 0)$, and can take a step towards the endpoint (T, N) either horizontally, vertically or diagonally, with the respective weights being $\frac{P_i}{2}$, P_d and $\frac{P_t}{2}$. This bears a close resemblance with the concept of a weighted Delannoy number [**? ? ? ?**] which is defined as follows.

Definition 4.2.1. *Consider a set of paths through a two-dimensional lattice that begin at* $(0, 0)$ *and end at* (r, s)*, where* $r, s \geq 0$*. At any point, these paths can either take a step* $(1, 0)$*,* $(0, 1)$ *or* $(1, 1)$ *with weights* α*,* β *or* γ *respectively. Then, a weighted Delannoy number* $d_{r,s}$ *is defined as the sum of weights of all possible paths from* $(0, 0)$ *to* (r, s)*. It can be characterized by the following recurrence relation:*

$$d_{r,s} = \alpha d_{r-1,s} + \beta d_{r,s-1} + \gamma d_{r-1,s-1} \tag{4.14}$$

and is subject to these initial conditions:

$$\begin{aligned} d_{0,0} &= 1 \\ d_{r,0} &= \alpha^r, \qquad r \geq 0 \\ d_{0,s} &= \beta^s, \qquad s \geq 0 \end{aligned}$$

On comparing (3.7) with (4.14), $P_T(\boldsymbol{y}_1^N)$ can evidently be seen as a weighted Delannoy number where the horizontal, vertical and diagonal edge weights are given by:

$$\begin{aligned} \alpha &= \frac{P_i}{2} \\ \beta &= P_d \\ \gamma &= \frac{P_t}{2} \end{aligned} \tag{4.15}$$

However, one key distinction in the computation of $P_T(\boldsymbol{y}_1^N)$ is that no horizontal steps are permitted in the last row of the lattice. This can easily be accounted for, by eliminating horizontal transitions in the final row, as follows:

$$P_T(\boldsymbol{y}_1^N) = d_{T,N} - \frac{P_i}{2} d_{T-1,N} \tag{4.16}$$

The above relation shows that in order to asymptotically approximate the probability of a received frame, it suffices to focus on weighted Delannoy numbers instead. In this regard, generating functions can serve as a powerful tool. It offers a way to represent an infinite sequence, by displaying its elements as coefficients of a formal power series. Since weighted Delannoy numbers form a bivariate sequence, the corresponding generating function would be of the form:

$$D(x,y) = \sum_{r,s \geq 0} d_{r,s} x^r y^s \tag{4.17}$$

Alternately, $D(x,y)$ can be transformed into a rational function in the following manner:

$$\begin{aligned} D(x,y) &= d_{0,0} + \sum_{r \geq 1} d_{r,0} x^r + \sum_{s \geq 1} d_{0,s} y^s + \sum_{r,s \geq 1} d_{r,s} x^r y^s \\ &= 1 + \sum_{r \geq 1} \beta^r x^r + \sum_{s \geq 1} \alpha^s y^s + \sum_{r,s \geq 1} d_{r,s} x^r y^s \\ &= 1 + \frac{\beta x}{1 - \beta x} + \frac{\alpha y}{1 - \alpha y} + \sum_{r,s \geq 1} d_{r,s} x^r y^s \end{aligned} \tag{4.18}$$

The final term in this equation may be simplified using (4.14):

$$\begin{aligned}
\sum_{r,s\geq 1} d_{r,s}x^r y^s &= \sum_{r,s\geq 1} (\alpha d_{r,s-1} + \beta d_{r-1,s} + \gamma d_{r-1,s-1})x^r y^s \\
&= \alpha \sum_{r,s\geq 1} d_{r,s-1}x^r y^s + \beta \sum_{r,s\geq 1} d_{r-1,s}x^r y^s + \gamma \sum_{r,s\geq 1} d_{r-1,s-1}x^r y^s \\
&= \alpha y \sum_{\substack{r\geq 1\\ s\geq 0}} d_{r,s}x^r y^s + \beta x \sum_{\substack{r\geq 0\\ s\geq 1}} d_{r,s}x^r y^s + \gamma xy \sum_{r,s\geq 1} d_{r-1,s-1}x^{r-1} y^{s-1} \\
&= \alpha y[D(x,y) - \sum_{\substack{r=0\\ s\geq 0}} d_{r,s}x^r y^s] + \beta x[f(x,y) - \sum_{\substack{r\geq 0\\ s=0}} d_{r,s}x^r y^s] + \gamma xy D(x,y) \\
&= \alpha y[D(x,y) - \sum_{s\geq 0} \alpha^s y^s] + \beta x[D(x,y) - \sum_{r\geq 0} \beta^r x^r] + \gamma xy D(x,y) \\
&= \alpha y[D(x,y) - \frac{1}{1-\alpha y}] + \beta x[D(x,y) - \frac{1}{1-\beta x}] + \gamma xy D(x,y) \\
&= (\alpha y + \beta x + \gamma xy)D(x,y) - \frac{\alpha y}{1-\alpha y} - \frac{\beta x}{1-\beta x} \qquad (4.19)
\end{aligned}$$

Using this result in (4.14),

$$\begin{aligned}
D(x,y) &= 1 + (\alpha y + \beta x + \gamma xy)D(x,y) \\
&= \frac{1}{1-\alpha y - \beta x - \gamma xy} \qquad (4.20)
\end{aligned}$$

For a univariate generating function, say $F(z) = \sum_{k\geq 0} f_k z^k$, one relatively simple way to extract asymptotic behavior of the coefficients is to use Cauchy's integral formula:

$$f_k = \frac{1}{2\pi i} \int_C \frac{F(z)}{z^{n+1}} dz \qquad (4.21)$$

where C refers to the contour of integration. The idea is to deform C such that it passes closely by the dominant singularities of $F(z)$, i.e. poles that lie closest to the origin. In doing so, most of the contribution to this integral is confined to a small neighborhood around the singularities. Hence to obtain an asymptotic approximation for this univariate sequence, it suffices to replace the integral in (4.21) with a sum of local expansions of the integrand around each singularity, via a Taylor's series.

Now to tackle the same problem for a bivariate generating function as in (4.20), we adopt the technique outlined in [**? ? ?**]. Without delving into much detail, it may be viewed as an extension of the aforementioned univariate approach, and is equipped to deal with

rational bivariate generating functions of the form:

$$F(x,y) = \frac{G(x,y)}{H(x,y)} \tag{4.22}$$

The first step involves determining the singular points of F along any specific direction in the first quadrant, say $\hat{\boldsymbol{r}}$. This is accomplished by solving the following equations:

$$\begin{aligned} H &= 0 \\ ry\frac{\partial H}{\partial y} &= sx\frac{\partial H}{\partial x} \end{aligned} \tag{4.23}$$

where (r,s) lies in the first quadrant along $\hat{\boldsymbol{r}}$. In regard to weighted Delannoy numbers, (4.23) translates to:

$$\begin{aligned} 1-\alpha y-\beta x-\gamma xy &= 0 \\ ry(\alpha+\gamma x) &= sx(\beta+\gamma y) \end{aligned} \tag{4.24}$$

The resulting solutions, formally known as critical points, are stated below:

$$\begin{aligned} (x_1,y_1) =& \left(\frac{2\omega-1-\frac{1}{\eta}+\frac{\rho'}{\eta}}{2\omega\beta}, \frac{2\omega-1-\eta+\rho'}{2\omega\alpha}\right) \\ (x_2,y_2) =& \left(\frac{2\omega-1-\frac{1}{\eta}-\frac{\rho'}{\eta}}{2\omega\beta}, \frac{2\omega-1-\eta-\rho'}{2\omega\alpha}\right) \end{aligned} \tag{4.25}$$

where

$$\begin{aligned} \omega &= \frac{\gamma}{\alpha\beta+\gamma} \\ \eta &= \frac{r}{s} \\ \rho' &= ((\eta+1)^2-4\omega\eta)^{1/2} \end{aligned} \tag{4.26}$$

From among these critical points, we need to extract the minimal point, which is essentially the singular point with the greatest impact on asymptotics of the sequence. In accordance with the results outlined in [**?**], when the concerned rational generating function is combinatorial in nature, i.e. has no negative coefficients in its power series expansion, as is the case here, it is sufficient to consider the critical point(s) that lie in the first quadrant alone. Furthermore, one of these critical points is guaranteed to be a minimal point. Upon numerical verification, it is found that (x_1,y_1) alone meets this criterion, and thus automatically qualifies as a minimal critical point. Due to the absence of other candidates, it can also be regarded as strictly minimal.
For the next step, we proceed to apply the following theorem from [**?**]:

Theorem 4.2.1. *Let $F = G/H$ be meromorphic and suppose that as $\hat{\boldsymbol{r}}$ varies in a neighborhood $\mathcal{N}$ of $\hat{\boldsymbol{r}_*}$ there is a smoothly varying, strictly minimal, smooth critical point $\boldsymbol{z}(\hat{\boldsymbol{r}_*})$ in direction $\hat{\boldsymbol{r}_*}$. Denoting $\boldsymbol{z} = (x, y)$ and $\boldsymbol{r} = (r, s)$, we suppose also that $G(\boldsymbol{z}(\hat{\boldsymbol{r}}))$ and the expression*

$$Q(\boldsymbol{z}(\hat{\boldsymbol{r}})) = -y^2 H_y^2 x H_x - y H_y x^2 H_x - x^2 y^2 (H_y^2 H_{xx} + H_x^2 H_{yy} - 2 H_x H_y H_{xy})$$

are nonzero for each $\hat{\boldsymbol{r}} \in \mathcal{N}$. Then as $|\boldsymbol{r}| \to \infty$, uniformly over $\hat{\boldsymbol{r}} \in \mathcal{N}$,

$$a_{\boldsymbol{r}} = (G + O(s^{-1/2})) \frac{1}{\sqrt{2\pi}} x^{-r} y^{-s} \sqrt{\frac{-y H_y}{sQ}} \tag{4.27}$$

In the following analysis, we consider that $D(x, y) = F(x, y)$. Now under the framework of Theorem 4.2.1, (x_1, y_1) is also a smooth point since partial derivatives of H do not vanish in its neighborhood. Hence, (x_1, y_1) fits the description of a strictly minimal, smooth critical point, and the aforementioned theorem is applicable. From (4.20) and (4.22), it follows that:

$$\begin{aligned} G &= 1 \\ H_x &= -(\beta + \gamma y) \\ H_y &= -(\alpha + \gamma x) \\ H_{xx} &= H_{yy} = 0 \\ H_{xy} &= -\gamma \end{aligned}$$

The above results lead to Q being:

$$\begin{aligned} Q(x, y) &= -x y^2 H_y^2 H_x - x^2 y H_y H_x^2 + 2 x^2 y^2 H_x H_y H_{xy} \\ &= -x y H_y H_x (y H_y + x H_x - 2 x y H_{xy}) \\ &= -x y (\alpha + \gamma x)(\beta + \gamma y)(-\alpha y - \gamma x y - \beta x - \gamma x y + 2 \gamma x y) \\ &= x y (\alpha + \gamma x)(\beta + \gamma y)(\alpha y + \beta x) \end{aligned} \tag{4.28}$$

Hence, the radical in (4.27) can be reduced to:

$$\begin{aligned} \frac{-y H_y}{sQ} &= \frac{y(\alpha + \gamma x)}{s x y (\alpha + \gamma x)(\beta + \gamma y)(\alpha y + \beta x)} \\ &= \frac{1}{s x (\beta + \gamma y)(\alpha y + \beta x)} \end{aligned} \tag{4.29}$$

As required in (4.27), we now proceed to evaluate the above expression at (x_1, y_1) by simplifying the denominator terms as follows:

$$\begin{aligned}
sx_1(\beta+\gamma y_1) &= \frac{s}{2\omega\beta}\big(2\omega-1-\frac{1}{\eta}+\frac{\rho'}{\eta}\big)\big(b+\frac{\gamma}{2\omega\alpha}(2\omega-1-\eta+\rho')\big)\\
&= \frac{s}{4\gamma^2\alpha\beta\eta}\big((2\omega-1)\eta-1+\rho'\big)\big(2\omega\alpha\beta+\gamma(2\omega-1-\eta+\rho')\big)\\
&= \frac{s}{4\omega^2\frac{\alpha\beta}{\alpha\beta+\gamma}\eta}\big((2\omega-1)\eta-1+\rho'\big)\big(\frac{2\omega\alpha\beta}{\alpha\beta+\gamma}+\frac{\gamma(2\omega-1-\eta+\rho')}{\alpha\beta+\gamma}\big)\\
&= \frac{s}{4\omega^2(1-\omega)\eta}\big((2\omega-1)\eta-1+\rho'\big)\big(2\omega(1-\omega)+\omega(2\omega-1-\eta+\rho')\big)\\
&= \frac{s}{4\omega(1-\omega)\eta}\big((2\omega-1)\eta-1+\rho'\big)\big(\rho'-\eta+1\big)\\
&= \frac{s}{4\omega(1-\omega)\eta}\big(2(\omega-1)\eta\rho'-(2\omega-1)\eta^2+2\omega\eta-1+\rho'^2\big)\\
&= \frac{s}{4\omega(1-\omega)\eta}\big(-2(1-\omega)\eta\rho'+2(1-\omega)\eta^2+2(1-\omega)\eta\big)\\
&= \frac{s}{2\omega}(\eta-\rho'+1) \qquad (4.30)\\
\alpha y_1+\beta x_1 &= \frac{1}{2\omega}\big(4\omega-2-\eta-\frac{1}{\eta}+\rho'(\eta+\frac{1}{\eta})\big)\\
&= \frac{1}{2\omega\eta}\big(4\omega\eta-2\eta-\eta^2-1+\rho'(1+\eta)\big)\\
&= \frac{1}{2\omega\eta}\big(-\rho'^2+\rho'(1+\eta)\big)\\
&= \frac{\rho'(\eta+1-\rho')}{2\omega\eta} \qquad (4.31)
\end{aligned}$$

Combining (4.29), (4.30) and (4.31):

$$\frac{-yH_y}{sQ} = \frac{\eta}{s\rho'}\left(\frac{2\omega}{\eta+1-\rho'}\right)^2 \qquad (4.32)$$

Finally using (4.32) in (4.27), the following asymptotic expression is reached:

$$d_{r,s} \sim \sqrt{\frac{\eta}{2\pi\rho' s}}\left(\frac{2\omega\beta}{2\omega-1+\frac{\rho'-1}{\eta}}\right)^r\left(\frac{2\omega a}{2\omega-1-\eta+\rho'}\right)^s\frac{2\omega}{\eta+1-\rho'} \qquad (4.33)$$

Recalling that (4.2) only requires us to evaluate the ratio of two coefficients, we can introduce further simplifications, based on the observation that their respective indices are relatively close. Thus we choose to approximate a ratio, say $\frac{d_{r_2,s_2}}{d_{r_1,s_1}}$, where $r_2 - r_1 = \delta_r << r_1$ and $s_2 - s_1 = \delta_s << s_1$.

To proceed along these lines, we first recognize that:

$$\begin{aligned}\lim_{r_1,s_1\to\infty}\frac{\eta_2}{\eta_1} &= \lim_{r_1,s_1\to\infty}\frac{r_1+\delta_r}{s_1+\delta_s}\cdot\frac{s_1}{r_1}\\ &= \lim_{r_1,s_1\to\infty}\frac{1+\frac{\delta_r}{r_1}}{1+\frac{\delta_s}{s_1}}\\ &= 1\end{aligned} \tag{4.34}$$

Additionally, since ρ' is a function of η, it may be written that:

$$\lim_{r_1,s_1\to\infty}\frac{\rho'_2}{\rho'_1} = 1 \tag{4.35}$$

Using (4.33), (4.34) and (4.35), we can conclude that:

$$\lim_{r_1,s_1\to\infty}\frac{d_{r_2,s_2}}{d_{r_1,s_1}} = \left(\frac{2\omega\beta}{2\omega-1+\frac{\rho'_1-1}{\eta_1}}\right)^{\delta_r}\left(\frac{2\omega\alpha}{2\omega-1-\eta_1+\rho'_1}\right)^{\delta_s} \tag{4.36}$$

The above result along with (4.16), allows us to deduce the following:

$$\begin{aligned}\lim_{r_1,s_1\to\infty}\frac{P_{r_2}(\boldsymbol{y}_1^{s_2})}{P_{r_1}(\boldsymbol{y}_1^{s_1})} &= \lim_{r_1,s_1\to\infty}\frac{d_{r_2,s_2}-\alpha d_{r_2,s_2-1}}{d_{r_1,s_1}-\alpha d_{r_1,s_1-1}}\\ &= \lim_{r_1,s_1\to\infty}\frac{d_{r_2,s_2}}{d_{r_1,s_1}}\cdot\frac{1-\frac{d_{r_2,s_2-1}}{d_{r_2,s_2}}}{1-\frac{d_{r_1,s_1-1}}{d_{r_1,s_1}}}\\ &= \lim_{r_1,s_1\to\infty}\frac{d_{r_2,s_2}}{d_{r_1,s_1}}\cdot\frac{1-\left(\frac{2\omega\alpha}{2\omega-1-\eta_2+\rho'_2}\right)^{-1}}{1-\left(\frac{2\omega\alpha}{2\omega-1-\eta_1+\rho'_1}\right)^{-1}}\\ &= \lim_{r_1,s_1\to\infty}\frac{d_{r_2,s_2}}{d_{r_1,s_1}}\\ &= \left(\frac{2\omega\beta}{2\omega-1+\frac{\rho'_1-1}{\eta_1}}\right)^{\delta_r}\left(\frac{2\omega\alpha}{2\omega-1-\eta_1+\rho'_1}\right)^{\delta_s}\\ &= x_1^{-\delta_r}y_1^{-\delta_s}\end{aligned} \tag{4.37}$$

Incorporating the preceding result in (4.2), we arrive at the following expression:

$$\begin{aligned}Z(\boldsymbol{v}_{t-1}\to\boldsymbol{v}_t) &= \log_2 P(\boldsymbol{m}^{bt}_{b(t-1)+1})+\log_2 P(\boldsymbol{y}^{ct+d_{ct}}_{c(t-1)+d_{c(t-1)}+1},d_{ct}|\boldsymbol{x}^{ct}_{c(t-1)+1},d_{c(t-1)})\\ &\quad -c\log_2\left(\frac{2\omega\beta}{2\omega-1+\frac{\rho'_1-1}{\eta_1}}\right)-(c+d_{ct}-d_{c(t-1)})\log_2\left(\frac{2\omega\alpha}{2\omega-1-\eta_1+\rho'_1}\right)\end{aligned}$$

Denoting branch output $\boldsymbol{x}_{c(t-1)+1}^{ct}$ as $\tilde{\boldsymbol{x}}_1^c$ and its slice of the received sequence $\boldsymbol{y}_{c(t-1)+d_{c(t-1)}+1}^{ct+d_{ct}}$ as $\tilde{\boldsymbol{y}}_1^{c+\delta}$, the above equation becomes:

$$\begin{aligned} Z(\boldsymbol{v}_{t-1} \to \boldsymbol{v}_t) &= \log_2 P(\boldsymbol{m}_{b(t-1)+1}^{bt}) + \log_2 P(\tilde{\boldsymbol{y}}_1^{c+\delta}, \delta | \tilde{\boldsymbol{x}}_1^c) \\ &\quad -c \log_2 \left(\frac{2\omega\beta}{2\omega - 1 + \frac{\rho_1' - 1}{\eta_1}} \right) - (c+\delta) \log_2 \left(\frac{2\omega\alpha}{2\omega - 1 - \eta_1 + \rho_1'} \right) \\ &= -b + \log_2 P(\tilde{\boldsymbol{y}}_1^{c+\delta}, \delta | \tilde{\boldsymbol{x}}_1^c) - c \log_2 \left(\frac{2\omega\beta}{2\omega - 1 + \frac{\rho_1' - 1}{\eta_1}} \right) \\ &\quad -(c+\delta) \log_2 \left(\frac{2\omega\alpha}{2\omega - 1 - \eta_1 + \rho_1'} \right) \end{aligned} \tag{4.38}$$

The second equality follows from the assumption that all messages are equi-probable. Notably, the preceding equation is now independent of the depth of branch $\boldsymbol{v}_{t-1} \to \boldsymbol{v}_t$ and instead is solely a function of $\tilde{\boldsymbol{x}}_1^c$ and $\tilde{\boldsymbol{y}}_1^{c+\delta}$. We may further simplify this by restricting our attention the asymptotic direction $\eta_1 = 1$. Such a constraint is fairly logical when insertion and deletion probabilities are comparable and a significant drift between the number of received and transmitted bits is not anticipated, even for longer frames.
Using (4.15), (4.26) and (4.25), coordinates of minimal point (x_1, y_1) can now be rewritten:

$$\begin{aligned} x_1 &= \frac{2\omega - 2 + 2\sqrt{1-\omega}}{2\omega\beta} \\ &= \frac{\sqrt{1-\omega} - (1-\omega)}{\omega P_d} \end{aligned} \tag{4.39}$$

$$\begin{aligned} y_1 &= \frac{2\omega - 2 + 2\sqrt{1-\omega}}{2\omega\alpha} \\ &= \frac{\sqrt{1-\omega} - (1-\omega)}{\omega \frac{P_i}{2}} \end{aligned} \tag{4.40}$$

where

$$\omega = \frac{\gamma}{\alpha\beta + \gamma} = \frac{P_t}{P_i P_d + P_t}$$

Finally, by incorporating (4.39) and (4.40) in (4.38), we arrive at the following asymptotic expression of a branch metric:

$$\begin{aligned} Z(\tilde{\boldsymbol{y}}_1^{c+\delta} | \tilde{\boldsymbol{x}}_1^c) &= -b + \log_2 P(\tilde{\boldsymbol{y}}_1^{c+\delta}, \delta | \tilde{\boldsymbol{x}}_1^c) - c \log_2 \left(\frac{\omega P_d}{\sqrt{1-\omega} - (1-\omega)} \right) \\ &\quad -(c+\delta) \log_2 \left(\frac{\omega \frac{P_i}{2}}{\sqrt{1-\omega} - (1-\omega)} \right) \end{aligned}$$

$$= -b + c + \delta + \log_2 P(\tilde{\boldsymbol{y}}_1^{c+\delta}, \delta | \tilde{\boldsymbol{x}}_1^c) - c \log_2 \left(\frac{\omega P_d}{\sqrt{1-\omega} - (1-\omega)} \right) - (c+\delta) \log_2 \left(\frac{\omega P_i}{\sqrt{1-\omega} - (1-\omega)} \right) \quad (4.41)$$

For the particular case when $P_i = P_d \neq 0$, we have:

$$\begin{aligned} \omega &= \frac{1 - P_d - P_i}{P_i P_d + 1 - P_i - P_d} \\ &= \frac{1 - 2P_i}{P_i^2 + 1 - 2P_i} \\ &= \frac{1 - 2P_i}{(1 - P_i)^2} \quad (4.42) \\ 1 - \omega &= \frac{P_i^2}{(1 - P_i)^2} \quad (4.43) \end{aligned}$$

As a consequence, the logarithmic arguments in (4.41) can be reduced to:

$$\begin{aligned} \frac{\omega P_d}{\sqrt{1-\omega} - (1-\omega)} &= \frac{\omega P_i}{\sqrt{1-\omega} - (1-\omega)} \\ &= \frac{(1-2P_i)P_i}{(\sqrt{1-\omega} - (1-\omega))(1-P_i)^2} \\ &= \frac{(1-2P_i)P_i}{(1-P_i)P_i - P_i^2} \\ &= \frac{1-2P_i}{1-P_i-P_i} \\ &= 1 \quad (4.44) \end{aligned}$$

Hence, when the insertion and deletion probabilities of a BSID channel are equal, the asymptotic expression of a branch metric is given by:

$$Z(\tilde{\boldsymbol{y}}_1^{c+\delta} | \tilde{\boldsymbol{x}}_1^c) = \log_2 P(\tilde{\boldsymbol{y}}_1^{c+\delta}, \delta | \tilde{\boldsymbol{x}}_1^c) + c + \delta - b \quad (4.45)$$

4.2.3 Computations in incorrect subtree

To resume the task of determining the average number of computations required per block, we refer again to (4.8). Here, the term $E[C(\boldsymbol{v}_0^*) | T_{\text{min}}^* = y]$ essentially refers to the expected number of computations needed to decode the first block correctly, given a particular value of T_{min}^*. From (4.4) we see that it can be decomposed into two terms: the number of visits to root $\boldsymbol{v}_0^*$, and nodes in the incorrect subtree τ_0. Recalling that the

code ensemble (c, b, ∞) imparts infinite depth to τ_0, we may write:

$$E[C(\boldsymbol{v}_0^*)|T_{\min}^* = y] \quad = \quad E[C'(\boldsymbol{v}_0^*)|T_{\min}^* = y] + \sum_{j=1}^{\infty} E\Big[\sum_{\boldsymbol{v}_j \in \tau_0} C'(\boldsymbol{v}_j)|T_{\min}^* = y\Big] \tag{4.46}$$

Now to determine the number of visits to a certain node, say $\boldsymbol{v}_j$, we note that in the first visit, the threshold T would be tightened around its metric $\mu(\boldsymbol{v}_j)$ as follows:

$$T \leq \mu(\boldsymbol{v}_j) < T + \Delta \tag{4.47}$$

Subsequently, $\boldsymbol{v}_j$ may be revisited with consecutively lower thresholds: $T-\Delta, T-2\Delta \ldots$. Let $T_1 = T - m\Delta$, where $m \in \mathbb{N}$, be the last threshold value with which $\boldsymbol{v}_j$ is visited. This would imply that the total number of visits to $\boldsymbol{v}_j$ is $m+1$, and (4.47) may be re-written as:

$$\begin{aligned} T_1 + m\Delta \leq \quad \mu(\boldsymbol{v}_j) \quad &< T_1 + (m+1)\Delta \\ m \leq \frac{\mu(\boldsymbol{v}_j) - T_1}{\Delta} &< m+1 \end{aligned}$$

Finally since $T_1 \geq T_{\min}^*$, the total number of visits to $\boldsymbol{v}_j$ can be upper-bounded in the following manner:

$$\begin{aligned} C'(\boldsymbol{v}_j) &= m+1 \\ &= \lfloor \frac{\mu(\boldsymbol{v}_j) - T_1}{\Delta} \rfloor + 1 \\ &\leq \frac{\mu(\boldsymbol{v}_j) - T_1}{\Delta} + 1 \\ &\leq \frac{\mu(\boldsymbol{v}_j) - T_{\min}^*}{\Delta} + 1 \end{aligned} \tag{4.48}$$

In order to obtain a more precise inequality, we introduce the following indicator function for any node $\boldsymbol{v}_j \in \tau_0$:

$$\phi(\boldsymbol{v}_j) = \begin{cases} 1, & \text{if node } \boldsymbol{v}_j \text{ is visited} \\ 0, & \text{else} \end{cases} \tag{4.49}$$

Since $\phi(\boldsymbol{v}_j) = 0$ naturally implies that $C'(\boldsymbol{v}_j) = 0$, we may combine (4.48) and (4.49):

$$C'(\boldsymbol{v}_j) \leq \left(\frac{\mu(\boldsymbol{v}_j - T_{\min}^*}{\Delta} + 1 \right) \phi(\boldsymbol{v}_j) \tag{4.50}$$

This inequality allows us to bound the number of visits to the root node $\boldsymbol{v}_0^*$ as follows:

$$C'(\boldsymbol{v}_0^*) \quad \leq \quad \frac{\mu(\boldsymbol{v}_0^*) - T_{\min}^*}{\Delta} + 1$$

$$= \quad -\frac{T^*_{\min}}{\Delta} + 1 \tag{4.51}$$

The second equality follows from the fact that metric of the root is always 0. Now, by applying (4.51) and (4.50) in (4.46), we get:

$$E[C(\boldsymbol{v}_0^*)|T^*_{\min} = y] \quad \leq \quad -\frac{y}{\Delta} + 1 + \sum_{j=1}^{\infty} E\Bigg[\sum_{\boldsymbol{v}_j \in \tau_0} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1\right)\phi(\boldsymbol{v}_j)\Bigg| T^*_{\min} = y\Bigg] \tag{4.52}$$

Equation (4.52) highlights the need to investigate the metric distribution of the nodes in τ_0, for the purpose of determining the net decoding effort wasted here. Equivalently, the behavior of branch metrics along these paths may be studied.

Referring back to Fig. 4.1, we observe that the nodes in τ_0 can be classified into two categories. One subset consists of nodes which hypothesize an entirely inaccurate message vector. Let this be denoted by τ_0'. The other set of nodes describes paths which point to the true transmitted frame, at least initially, but suggest a different drift sequence. We refer to this subset as τ_0^*. For the nodes in τ_0', it is reasonable to assume that the hypothesized output block is statistically independent of the received block. However for the nodes in τ_0^* which wholly or partially indicate the true input, such an assumption is clearly invalid, as the predicted output does depend on what was transmitted over the channel. This suggests that the node metrics in τ_0' and τ_0^* do not follow a similar distribution, and thus, we choose to analyze them separately.

$$\begin{aligned} &\sum_{j=1}^{\infty} E\Bigg[\sum_{\boldsymbol{v}_j \in \tau_0} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1\right)\phi(\boldsymbol{v}_j)\Bigg| T^*_{\min} = y\Bigg] \\ = \quad &\sum_{j=1}^{\infty} E\Bigg[\sum_{\boldsymbol{v}_j \in \tau_0'} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1\right)\phi(\boldsymbol{v}_j)\Bigg| T^*_{\min} = y\Bigg] \\ &+ \sum_{j=1}^{\infty} E\Bigg[\sum_{\boldsymbol{v}_j \in \tau_0^*} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1\right)\phi(\boldsymbol{v}_j)\Bigg| T^*_{\min} = y\Bigg] \end{aligned} \tag{4.53}$$

4.2.4 On incorrect paths

We begin by addressing the former term in (4.53), which signifies the average number of computations performed in τ_0', for a given value of minimum threshold. As stated earlier, the paths in τ_0' predict a transmitted frame that is wholly inaccurate, and may be regarded as statistically independent of the received frame. To proceed with determining the average number of computations performed on these paths, we need to study the distribution of node metrics in this subset. Equivalently, we may also choose to examine how the branch metrics behave along these paths, and given their dependence on relative drifts, their probability distributions should be considered in a drift-specific manner. To

this end, we let a random variable $Z'(\delta)$ describe the probability distribution of branch metrics in τ_0', for a specific drift change δ.
Before starting our analysis under this framework, we first consider the basic case of the nodes in τ_0' at depth 1. From Fig. 4.1, we can deduce that among the branches that emerge from the root node, $(2^b-1)(i_{\max}+d_{\max}+1)$ of them correspond to a different input from the correct path p^*. Thus, we may write:

$$\begin{aligned} & E\Big[\sum_{\boldsymbol{v}_1\in\tau_0'}\Big(\frac{\mu(\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)\Big|T^*_{\min}=y\Big] \\ = \;& E\Big[\sum_{\boldsymbol{v}_1\in\tau_0'}\Big(\frac{Z(\boldsymbol{v}_0\to\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)\Big|T^*_{\min}=y\Big] \\ = \;& (2^b-1)\sum_{\delta=-d_{\max}}^{i_{\max}}\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z'(\delta)=z_1) \end{aligned} \tag{4.54}$$

It is possible to simplify the above expression by subsuming the drift-specific probability distributions under a single random variable, say ζ'. It is meant to act as a uniform mixture of the random variable $Z'(-d_{\max}),\cdots Z'(i_{\max})$. More explicitly, the probability function of ζ' combines those of $Z'(-d_{\max}),\cdots Z'(i_{\max})$ as follows:

$$P(\zeta'=z)=\frac{1}{\lambda}\sum_{\delta=-d_{\max}}^{i_{\max}}P(Z'(\delta)=z) \tag{4.55}$$

where

$$\lambda=i_{\max}+d_{\max}+1$$

This helps us do away with the drift-specific probability distributions in (4.54), thereby allowing us to represent the branch metrics with a single random variable ζ', as demonstrated in Fig. 4.2.
Using (4.55), we can now simplify (4.54) as follows:

$$\begin{aligned} & E\Big[\sum_{\boldsymbol{v}_1\in\tau_0'}\Big(\frac{\mu(\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)\Big|T^*_{\min}=y\Big] \\ = \;& (2^b-1)\sum_{\delta=-d_{\max}}^{i_{\max}}\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z'(\delta)=z_1) \\ = \;& (2^b-1)\lambda\sum_{\delta=-d_{\max}}^{i_{\max}}\sum_{z_1\geq y}\frac{1}{\lambda}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z'(\delta)=z_1) \end{aligned}$$

$$= (2^b - 1)\lambda \sum_{z_1 \geq y} \left(\frac{z_1 - y}{\Delta} + 1 \right) \frac{1}{\lambda} \sum_{\delta = -d_{\max}}^{i_{\max}} P(Z'(\delta) = z_1)$$

$$= (2^b - 1)\lambda \sum_{z_1 \geq y} \left(\frac{z_1 - y}{\Delta} + 1 \right) P(\zeta' = z_1) \tag{4.56}$$

Before attempting to generalize this equation to longer paths, we recognize that a node

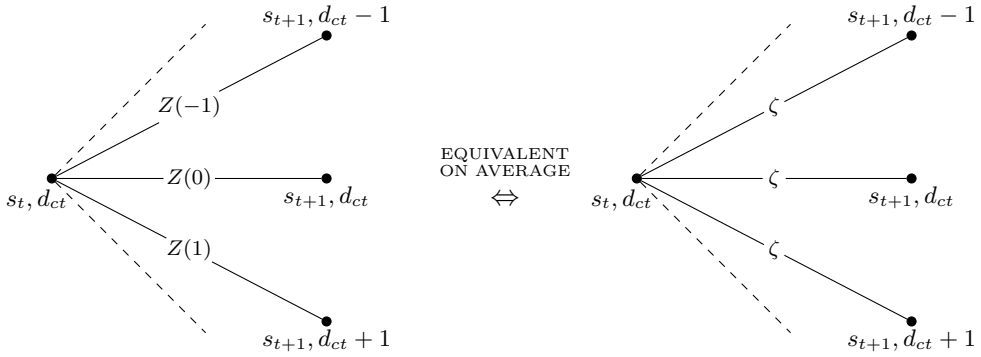


Figure 4.2: Characterization of branch metrics

at depth j, say $\boldsymbol{v}_j$, is eligible for a visit by the decoder only if the metric values of $\boldsymbol{v}_j$ and its predecessors are greater than or equal to the minimum threshold $T^*_{\min}$. Stated differently, branch metrics along the path described by $\boldsymbol{v}_j$ should uphold the following conditions:

$$\sum_{i=1}^{k} z_i \geq T^*_{\min}, \quad \forall \ 1 \leq k \leq j \tag{4.57}$$

where z_i refers to the metric value of the i^{th} branch along $\boldsymbol{v}_j$.
Let $\boldsymbol{\mathcal{Z}}^{(j)}(y) \in \mathbb{R}^j$ denote the set of all vectors $\boldsymbol{z}$ that uphold (4.57) for a specific $T^*_{\min} = y$. Under this construct, we can extend (4.56) to further depths in the following manner:

$$\sum_{\boldsymbol{v}_j \in \tau'_0} E\left[\left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_j) \middle| T^*_{\min} = y \right]$$

$$= (2^b - 1)2^{b(j-1)} \sum_{\boldsymbol{\delta} = (\delta_1, \cdots, \delta_j)} \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1 \right) P(Z'_1(\delta_1) = z_1, \cdots Z'_j(\delta_j) = z_j) \tag{4.58}$$

Assuming that branch metrics $Z'(\delta)$ are independently distributed, we may write:

$$\begin{aligned}\sum_{\boldsymbol{\delta}=(\delta_1,\cdots,\delta_j)} P(Z'_1(\delta_1)=z_1,\cdots Z'_j(\delta_j)=z_j) &= \sum_{\boldsymbol{\delta}=(\delta_1,\cdots,\delta_j)} \prod_{i=1}^{j} P(Z'_i(\delta_i)=z_i) \\ &= \lambda^j \prod_{i=1}^{j} P(\zeta'_i=z_i) \end{aligned} \tag{4.59}$$

The preceding equation enables us to simplify (4.58) as follows:

$$\begin{aligned}&\sum_{\boldsymbol{v}_j\in\tau'_0} E\left[\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\middle| T^*_{\min}=y\right] \\ =\;& (2^b-1)2^{b(j-1)}\lambda^j \sum_{\boldsymbol{\delta}=(\delta_1,\cdots,\delta_j)} \sum_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i-y}{\Delta}+1\right)\prod_{i=1}^{j}\frac{1}{\lambda}P(Z'_i(\delta_i)=z_i) \\ =\;& (2^b-1)2^{b(j-1)}\lambda^j \sum_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i-y}{\Delta}+1\right) \sum_{\boldsymbol{\delta}=(\delta_1,\cdots,\delta_j)} \prod_{i=1}^{j}\frac{1}{\lambda}P(Z'_i(\delta_i)=z_i) \\ =\;& (2^b-1)2^{b(j-1)}\lambda^j \sum_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i-y}{\Delta}+1\right)\prod_{i=1}^{j} P(\zeta'_i=z_i) \end{aligned} \tag{4.60}$$

Since the maximum number of visits to any node, if visited at least once, is ultimately dependent on its cumulative metric value alone, it might be more convenient to redefine the paths in τ'_0 in terms of their respective cumulative node metrics. To rephrase, any node $\boldsymbol{v}_j$ described by a branch metric vector $\boldsymbol{z}=(z_1,\dots z_j)$ can also be represented by:

$$\boldsymbol{\mu}=(\mu_1,\mu_2,\dots\mu_j)$$

where $\mu_k=\sum_{i=1}^{k} z_i$ is the metric value of the k^{th} node along $\boldsymbol{v}_j$. Let M_j represent a random variable denoting the node metric along a path in τ'_0 at depth j. Furthermore, we define a set $\boldsymbol{\mathcal{M}}^{(j)}(y)$, analogous to $\boldsymbol{\mathcal{Z}}^{(j)}(y)$, such that for each $\boldsymbol{\mu}\in\boldsymbol{\mathcal{M}}^{(j)}(y)$:

$$\mu_k\geq y,\quad \forall\ 1\leq k\leq j \tag{4.61}$$

This allows us to reformulate (4.60) as:

$$\sum_{\boldsymbol{v}_j\in\tau'_0} E\left[\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\middle| T^*_{\min}=y\right]$$

$$
\begin{aligned}
&= (2^b-1)2^{b(j-1)}\lambda^j \sum_{\boldsymbol{z}\in\boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta}+1\right)\prod_{i=1}^{j} P(\zeta_i'=z_i) \\
&= (2^b-1)2^{b(j-1)}\lambda^j \sum_{\boldsymbol{\mu}\in\boldsymbol{\mathcal{M}}^{(j)}(y)} \left(\frac{\mu_j - y}{\Delta}+1\right) P(M_1'=\mu_1,\cdots M_j'=\mu_j) \\
&= (2^b-1)2^{b(j-1)}\lambda^j \sum_{\mu_j\geq y} \left(\frac{\mu_j - y}{\Delta}+1\right) P(M_1'\geq y,\cdots M_{j-1}'\geq y, M_j'=\mu_j) \quad (4.62)
\end{aligned}
$$

For notational convenience, we introduce a quantity $f_j(y,z)$:

$$
f_j(y,\mu) = P(M_1'\geq y,\cdots M_{j-1}'\geq y, M_j'=\mu) \quad (4.63)
$$

Hence, (4.60) finally becomes:

$$
\begin{aligned}
&\sum_{\boldsymbol{v}_j\in\tau_0'} E\left[\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\middle| T_{\min}^{*}=y\right] \\
&= (2^b-1)2^{b(j-1)}\lambda^j \sum_{\mu\geq y}\left(\frac{\mu-y}{\Delta}+1\right) f_j(y,\mu) \quad (4.64)
\end{aligned}
$$

We can now combine (4.56) and (4.64) to arrive at an upper bound for the total number of visits to all nodes in τ_0':

$$
\begin{aligned}
&\sum_{j=1}^{\infty} E\left[\sum_{\boldsymbol{v}_j\in\tau_0'}\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\middle| T_{\min}^{*}=y\right] \\
&= (2^b-1)\sum_{j=1}^{\infty} 2^{b(j-1)}\lambda^j \sum_{\mu\geq y}\left(\frac{\mu-y}{\Delta}+1\right) f_j(y,\mu) \\
&= (1-2^{-b})\sum_{j=1}^{\infty} 2^{bj}\lambda^j \sum_{\mu\geq y}\left(\frac{\mu-y}{\Delta}+1\right) f_j(y,\mu) \\
&= (1-2^{-b})\left(\sum_{j=1}^{\infty} 2^{b'j}\sum_{\mu\geq y} f_j(y,\mu) + \Delta^{-1}\sum_{j=1}^{\infty} 2^{b'j}\sum_{\mu\geq y}(\mu-y) f_j(y,\mu)\right) \quad (4.65)
\end{aligned}
$$

where $b' = b + \log_2\lambda$.

We first aim to acquire an upper bound on the former term, which is roughly represents the number of unique nodes visited in τ_0'. In order to accomplish this, we recognize that $\sum_{\mu\geq y} f_j(y,\mu)$ simply refers to the probability that during a random walk, all nodes traversed upto depth j, have metric values which are still above or equal to y. However, since the expected branch metric along an incorrect path is typically negative, any infinite

random walk in τ_0' will eventually fall below a finite y. Thus it becomes possible to state that:

$$\sum_{t=1}^{\infty}\sum_{z<y} f_t(y,z) = 1 \tag{4.66}$$

Consequently, the probability that an infinite random walk has not yet crossed the barrier y at depth j, or $\sum_{z\geq y} f_j(y,z)$, can also be viewed as the probability that this random walk crosses the barrier at a depth beyond j. This equivalence can be expressed mathematically as follows:

$$\sum_{z\geq y} f_j(y,z) = \sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \tag{4.67}$$

In the above relation, $\sum_{\mu<y} f_{j+1}(y,\mu)$ denotes the probability that a node at depth j, unlike all of its predecessors, has a metric value lower than threshold y. By incorporating (4.67) in (4.65), we arrive at:

$$\begin{aligned}
\sum_{j=1}^{\infty} 2^{b'j}\sum_{\mu\geq y} f_j(y,\mu) &= \sum_{j=1}^{\infty} 2^{b'j}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \\
&= \frac{1}{1-2^{-b'}}\sum_{j=1}^{\infty}(2^{b'j}-2^{b'(j-1)})\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \\
&= \frac{1}{1-2^{-b'}}\Bigg(\sum_{j=1}^{\infty} 2^{b'j}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \\
&\quad -\sum_{j=1}^{\infty} 2^{b'(j-1)}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu)\Bigg) \\
&= \frac{1}{1-2^{-b'}}\Bigg(\sum_{t=1}^{\infty}\sum_{z<y} f_t(y,z) - 1 + \sum_{j=1}^{\infty} 2^{b'j}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \\
&\quad -\sum_{j=1}^{\infty} 2^{b'(j-1)}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu)\Bigg)
\end{aligned}$$

The final equality follows from (4.66). We can continue the simplification as follows:

$$\begin{aligned}
\sum_{j=1}^{\infty} 2^{b'j}\sum_{\mu\geq y} f_j(y,\mu) &= \frac{1}{1-2^{-b'}}\Bigg(\sum_{t=1}^{\infty}\sum_{z<y} f_t(y,z) + \sum_{j=1}^{\infty} 2^{b'j}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) \\
&\quad -\sum_{j=0}^{\infty} 2^{b'j}\sum_{t=j+2}^{\infty}\sum_{\mu<y} f_t(y,\mu) - 1\Bigg) \\
&= \frac{1}{1-2^{-b'}}\Bigg(\sum_{j=0}^{\infty} 2^{b'j}\sum_{t=j+1}^{\infty}\sum_{\mu<y} f_t(y,\mu) - \sum_{j=0}^{\infty} 2^{b'j}\sum_{t=j+2}^{\infty}\sum_{\mu<y} f_t(y,\mu) - 1\Bigg)
\end{aligned}$$

$$= (1-2^{-b'})^{-1}\Big(\sum_{j=0}^{\infty} 2^{b'j} \sum_{\mu<y} f_{j+1}(y,\mu) - 1\Big) \tag{4.68}$$

Hence, the number of unique nodes in τ_0' that stay above threshold, is proportional to the number of nodes which just drop below it. Now, to obtain an upper bound on this quantity, we consider a random walk p' along any path in τ_0', with the consecutive branch metrics being ζ_0', ζ_1' and so on. As before, ζ_j''s are assumed to be independent and identically distributed.

We also define a moment generating function for branch metrics along an incorrect path, $g_1(\sigma)$ as follows:

$$\begin{aligned} g_1(\sigma) &= E[2^{\sigma\zeta'}] \\ &= \sum_z P(\zeta'=z)2^{\sigma z} \end{aligned} \tag{4.69}$$

where $P(\zeta'=z)$ is given by (4.55). The expected value of this branch metric can also be computed from (4.69):

$$E[\zeta'] = \frac{g_1'(\sigma)}{\ln 2}\Big|_{\sigma=0} \tag{4.70}$$

The above quantity essentially represents the expected branch metric along any incorrect

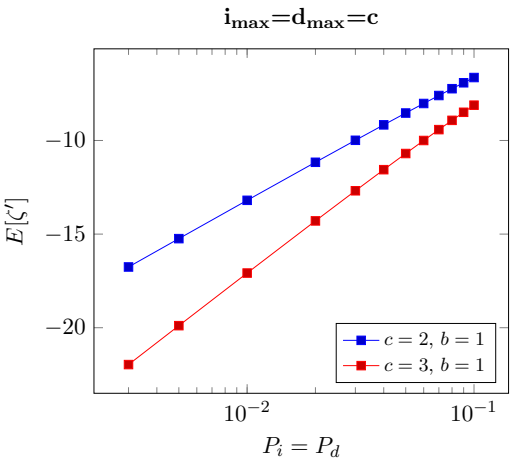


Figure 4.3: Expected branch metric along incorrect paths

path. For fairly low error probabilities, $E[\zeta']$ is typically negative, as illustrated in Fig. 4.3. In fact, Fano's algorithm relies on its negativity, since otherwise the decoder would be likely to pick the wrong successor at every step, and ultimately offer an incorrect estimate for the transmitted frame. Now instead of using the distribution $P(\zeta'=z)$, we

consider the following tilted probability assignment:

$$P(\zeta'_\sigma = z) = \frac{2^{\sigma z} P(\zeta' = z)}{g_1(\sigma)} \tag{4.71}$$

This probability distribution is clearly valid since $P(\zeta'_\sigma = z) \geq 0$ for all z, and

$$\begin{aligned} \sum_z P(\zeta'_\sigma = z) &= \frac{1}{g_1(\sigma)} \sum_z 2^{\sigma z} P(\zeta' = z) \\ &= 1 \end{aligned}$$

Analogous to p', we consider another random walk p'_σ, with the corresponding branch metrics being $\zeta'_{0,\sigma}, \zeta'_{1,\sigma}, \ldots$. These metric values are guided by (4.71) and we may thus write:

$$P(\zeta'_{i,\sigma} = a) = \frac{P(\zeta'_i = a) 2^{\sigma a}}{g_1(\sigma)}$$

where the random walk $\zeta'_0, \zeta'_1, \ldots$ obey the probability distribution (4.55), as in the case of p'. Using the above relation in (4.63), we get:

$$\begin{aligned} f_{\sigma,j}(y,\mu) &= P(M'_{1,\sigma} \geq y, \cdots, M'_{j-1,\sigma} \geq y, M'_{j,\sigma} = \mu) \\ &= \sum_{\boldsymbol{\mu}_1^{(j-1)} \in \boldsymbol{\mathcal{M}}^{(j-1)}(y)} P(M'_{1,\sigma} = \mu_1, \cdots, M'_{j-1,\sigma} = \mu_{j-1}, M'_{j,\sigma} = \mu) \\ &= \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} P(\zeta'_{1,\sigma} = z_1, \cdots, \zeta'_{j-1,\sigma} = z_{j-1}, \zeta'_{j,\sigma} = \mu - \sum_{i=1}^{j-1} z_i) \end{aligned}$$

where $\boldsymbol{\mathcal{Z}}^{(j-1)}(y)$ and $\boldsymbol{\mathcal{M}}^{(j-1)}(y)$ follow their prior definitions as per (4.57) and (4.61) respectively. With the assumption of independence of the branch metrics $Z_{i,\sigma}$, we can continue simplifying the previous equation:

$$\begin{aligned} f_{\sigma,j}(y,\mu) &= \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} \left(\prod_{i=1}^{j-1} P(\zeta'_{i,\sigma} = z_i) \right) P(\zeta'_{j,\sigma} = \mu - \sum_{i=1}^{j-1} z_i) \\ &= g_1(\sigma)^{-j} \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} \left(\prod_{i=1}^{j-1} P(\zeta'_i = z_i) 2^{\sigma z_i} \right) P(\zeta'_j = \mu - \sum_{i=1}^{j-1} z_i) 2^{\sigma(\mu - \sum_{i=1}^{j-1} z_i)} \\ &= g_1(\sigma)^{-j} 2^{\sigma\mu} \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} \left(\prod_{i=1}^{j-1} P(\zeta'_i = z_i) \right) P(\zeta'_j = \mu - \sum_{i=1}^{j-1} z_i) \end{aligned}$$

$$
\begin{aligned}
&= g_1(\sigma)^{-j} 2^{\sigma\mu} \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} \Big(\prod_{i=1}^{j-1} P(\zeta_i' = z_i) \Big) P(\zeta_j' = \mu - \sum_{i=1}^{j-1} z_i) \\
&= g_1(\sigma)^{-j} 2^{\sigma\mu} \sum_{\boldsymbol{z}_1^{(j-1)} \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y)} P(\zeta_1' = z_1, \cdots, \zeta_{j-1}' = z_{j-1}, \zeta_j' = \mu - \sum_{i=1}^{j-1} z_i) \\
&= g_1(\sigma)^{-j} 2^{\sigma\mu} \sum_{\boldsymbol{\mu}_1^{(j-1)} \in \boldsymbol{\mathcal{M}}^{(j-1)}(y)} P(M_1' = \mu_1, \cdots, M_{j-1}' = \mu_{j-1}, M_j' = \mu) \\
&= g_1(\sigma)^{-j} 2^{\sigma\mu} P(M_1' \geq y, \cdots, M_{j-1}' \geq y, M_j' = \mu) \\
&= f_{0,j}(y,\mu) 2^{\sigma\mu} g_1(\sigma)^{-j} \qquad (4.72)
\end{aligned}
$$

If σ is so chosen that $E[\zeta_\sigma'] = \sum_{z'} z' P(\zeta_\sigma' = z')$ remains negative as it does for an unbiased random walk in (4.70), it becomes possible to restate (4.66) as follows:

$$
\sum_{t=1}^{\infty} \sum_{\mu<y} f_{\sigma,t}(y,\mu) = \sum_{t=1}^{\infty} \sum_{\mu<y} f_{0,t}(y,\mu) 2^{\sigma\mu} g_1(\sigma)^{-t} = 1 \qquad (4.73)
$$

Choosing $\sigma_1 > 0$ such that $g_1(\sigma_1) = 2^{-b'}$, the above equation transforms to:

$$
\sum_{t=1}^{\infty} \sum_{\mu<y} f_{\sigma_1,t}(y,\mu) = \sum_{t=1}^{\infty} \sum_{\mu<y} f_{0,t}(y,\mu) 2^{\sigma_1\mu} 2^{b't} = 1 \qquad (4.74)
$$

Referring back to the definition of $f_j(y,\mu)$ in (4.63),

$$
\sum_{\mu<y} f_{0,t}(y,\mu) 2^{\sigma_1\mu} = \sum_{\mu<y} P(M_1' \geq y, \cdots M_{t-1}' \geq y, M_t' = \mu) 2^{\sigma_1\mu}
$$

Now if $M_{t-1}' \geq y$, it holds that $M_t' = M_{t-1}' + \zeta_t' \geq y + z_{\min}$, where $z_{\min}' < 0$ denotes the minimum value that ζ' may assume. Thus the preceding expression can be lower-bounded in the following manner:

$$
\sum_{\mu<y} f_{0,t}(y,\mu) 2^{\sigma_1\mu} \geq 2^{\sigma_1(y+z_{\min}')} \sum_{\mu<y} f_{0,t}(y,\mu) \qquad (4.75)
$$

By combining (4.74) and (4.75), we can conclude that:

$$
\begin{aligned}
2^{\sigma_1(y+z_{\min})} \sum_{t=1}^{\infty} 2^{b't} \sum_{\mu<y} f_{0,t}(y,\mu) &\leq \sum_{t=1}^{\infty} \sum_{\mu<y} f_{0,t}(y,\mu) 2^{\sigma_1\mu} 2^{b't} = 1 \\
\implies \sum_{t=1}^{\infty} 2^{b't} \sum_{\mu<y} f_{0,t}(y,\mu) &\leq 2^{-\sigma_1(y+z_{\min})} \qquad (4.76)
\end{aligned}
$$

Using this inequality in (4.68), we conclude that:

$$\begin{aligned}\sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu \geq y} f_j(y,\mu) &= (1-2^{-b'})^{-1}\Big(\sum_{j=0}^{\infty} 2^{b'j} \sum_{\mu<y} f_{j+1}(y,\mu) - 1\Big) \\ &= (1-2^{-b'})^{-1}\Big(\sum_{j=1}^{\infty} 2^{b'(j-1)} \sum_{\mu<y} f_j(y,\mu) - 1\Big) \\ &\leq \frac{2^{-\sigma_1(y+z_{\min})-b'}-1}{1-2^{-b'}} \end{aligned} \tag{4.77}$$

Hence, we can finally establish an upper bound on the number of nodes in τ_0' that do not fall below the threshold y:

$$(1-2^{-b})\sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu \geq y} f_j(y,\mu) \leq \frac{1-2^{-b}}{1-2^{-b'}}\big(2^{-\sigma_1(y+z_{\min})-b'}-1\big) \tag{4.78}$$

To resume the task of bounding the decoding effort expended on paths in τ_0', we now focus on the latter term in (4.65). In order to bound this quantity, we observe that on differentiating (4.73) with respect to σ, we get

$$\sum_{j=1}^{\infty}\sum_{\mu<y} f_{0,j}(y,\mu)\mu 2^{\sigma\mu} g_1(\sigma)^{-j} \ln 2 - \sum_{j=1}^{\infty}\sum_{\mu<y} f_{0,j}(y,\mu) 2^{\sigma\mu} j g_1(\sigma)^{-j-1} g_1'(\sigma) = 0$$

Using (4.70), we can rewrite the above relation as:

$$\sum_{j=1}^{\infty}\sum_{\mu<y} f_{0,j}(y,\mu)\mu 2^{\sigma\mu} g_1(\sigma)^{-j} = \sum_{j=1}^{\infty}\sum_{\mu<y} f_{0,j}(y,\mu) 2^{\sigma\mu} j g_1(\sigma)^{-j-1} E[\zeta']$$

Upon evaluating this at $\sigma = 0$, we deduce that:

$$\begin{aligned}\sum_{j=1}^{\infty}\sum_{\mu<y} f_{0,j}(y,\mu)\mu &= E[\zeta']\sum_{j=1}^{\infty} j \sum_{\mu<y} f_{0,j}(y,\mu) \\ E[\mu_N' | T_{\min}^* = y] &= E[\zeta']E[N|T_{\min}^* = y] \end{aligned} \tag{4.79}$$

where N is a random variable denoting the depth in a code tree where node metric is expected to drop below threshold $T_{\min}^* = y$ for the first time. Since $y + z_{\min} \leq E[\mu_N'|T_{\min}^* = y] \leq y$,

$$y \geq E[\zeta']E[N|T_{\min}^* = y]$$

$$= E[\zeta'] \sum_{j=1}^{\infty} j \sum_{\mu<y} f_{0,j}(y,\mu) \tag{4.80}$$

Equivalently,

$$\begin{aligned} y-\mu &\geq E[\zeta'] \sum_{j=1}^{\infty} j \sum_{x<y-\mu} f_j(y-\mu,x) \\ \implies \mu-y &\leq -E[\zeta'] \sum_{j=1}^{\infty} j \sum_{x<y-\mu} f_j(y-\mu,x) \end{aligned}$$

This now assists in upper-bounding the second term in (4.65) as follows:

$$\begin{aligned} \sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu\geq y} (\mu-y) f_j(y,\mu) &\leq -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu\geq y} \sum_{t=1}^{\infty} t \sum_{x<y-\mu} f_t(y-\mu,x) f_j(y,\mu) \\ &= -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu\geq y} \sum_{t=1}^{\infty} t f_j(y,\mu) \sum_{x<y-\mu} f_t(y-\mu,x) \\ &= -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{t=1}^{\infty} t \sum_{\mu\geq y} f_j(y,\mu) \sum_{x<y-\mu} f_t(y-\mu,x) \end{aligned} \tag{4.81}$$

To gain an insight into the term $\sum_{\mu\geq y} f_j(y,\mu) \sum_{x<y-\mu} f_t(y-\mu,x)$ we consider the following:

- Component $f_j(y,\mu)$ suggests that at depth j, the node metric is μ, such that $\mu > y$, i.e barrier hasn't been crossed.
- The term $\sum_{x<y-\mu} f_t(y-\mu,x)$ in isolation, refers to the probability that a random walk with root node metric 0, crosses the barrier $y-\mu$ at depth t. Equivalently, it may also be seen as the probability that a random walk, where the starting node metric is z, crosses barrier y at depth t.

By combining these observations, we may view the composite term $f_j(y,\mu) f_t(y-\mu,x)$ as the probability that during a random walk, metric values at depth j and $j+t$ are μ and $x+\mu$ respectively. We may extend this interpretation further and view $\sum_{\mu\geq y} f_j(y,\mu) \sum_{x<y-\mu} f_t(y-\mu,x)$ as the probability that during a random walk, metric value at depth j stays above the barrier y, but metric at depth $j+t$ falls below y. To put it more concisely,

$$\sum_{\mu\geq y} f_j(y,\mu) \sum_{x<y-\mu} f_t(y-\mu,x) = \sum_{\mu<y} f_{j+t}(y,\mu) \tag{4.82}$$

since by the definition of $f_{j+t}(y,\mu)$ in (4.63), it is already guaranteed that nodes prior to depth $j+t$ have metric values lying above the barrier y. By applying this simplification in (4.81), we obtain the following:

$$\begin{aligned}
\sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu \geq y} (\mu - y) f_j(y,\mu) &\leq -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{t=1}^{\infty} t \sum_{\mu \geq y} f_j(y,\mu) \sum_{x < y-\mu} f_t(y-\mu, x) \\
&= -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{t=1}^{\infty} t \sum_{\mu < y} f_{j+t}(y,\mu) \\
&= -E[\zeta'] \sum_{j=1}^{\infty} 2^{b'j} \sum_{k-j=1}^{\infty} (k-j) \sum_{\mu < y} f_k(y,\mu) \\
&= -E[\zeta'] \sum_{k=1}^{\infty} \sum_{j=1}^{k} (k-j) 2^{b'j} \sum_{\mu < y} f_k(y,\mu) \\
&= -E[\zeta'] \sum_{k=1}^{\infty} \sum_{\mu < y} f_k(y,\mu) \sum_{j=1}^{k} (k-j) 2^{b'j} \qquad (4.83)
\end{aligned}$$

The term $\sum_{j=1}^{k} (k-j) 2^{b'j}$ can be expanded in the following manner:

$$\begin{aligned}
\sum_{j=1}^{k} (k-j) 2^{b'j} &= k \sum_{j=1}^{k} 2^{b'j} - \sum_{j=1}^{k} j 2^{b'j} \\
&= k 2^{b'} \frac{2^{b'k} - 1}{2^{b'} - 1} - \frac{k 2^{b'(k+2)} - 2^{b'(k+1)}(k+1) + 2^{b'}}{(2^{b'} - 1)^2} \\
&= \frac{-k 2^{b'}}{2^{b'} - 1} + \frac{2^{b'(k+1)}}{(2^{b'} - 1)^2} - \frac{2^{b'}}{(2^{b'} - 1)^2} \qquad (4.84)
\end{aligned}$$

Incorporating the above result in (4.83),

$$\begin{aligned}
\sum_{j=1}^{\infty} 2^{b'j} \sum_{\mu \geq y} (\mu - y) f_j(y,z) &\leq \frac{2^{b'}}{2^{b'} - 1} E[\zeta'] \sum_{k=1}^{\infty} k \sum_{\mu < y} f_k(y,\mu) \\
&\quad - \frac{2^{b'}}{(2^{b'} - 1)^2} E[\zeta'] \sum_{k=1}^{\infty} 2^{b'k} \sum_{\mu < y} f_k(y,\mu) \\
&\quad + \frac{2^{b'}}{(2^{b'} - 1)^2} E[\zeta'] \sum_{k=1}^{\infty} \sum_{\mu < y} f_k(y,\mu) \\
&= \frac{2^{b'}}{2^{b'} - 1} E[\zeta'] \sum_{k=1}^{\infty} k \sum_{\mu < y} f_k(y,\mu) + \frac{2^{b'}}{(2^{b'} - 1)^2} E[\zeta']
\end{aligned}$$

$$-\frac{2^{b'}}{(2^{b'}-1)^2}E[\zeta']\sum_{k=1}^{\infty}2^{b'k}\sum_{\mu<y}f_k(y,\mu) \tag{4.85}$$

From (4.70),(4.76) and (4.80), we can reduce (4.85) to:

$$\begin{aligned}\sum_{j=1}^{\infty}2^{b'j}\sum_{\mu\geq y}(\mu-y)f_j(y,z) &\leq \frac{2^{b'}}{2^{b'}-1}y-\frac{2^{b'-\sigma_1(y+z_{\min})}}{(2^{b'}-1)^2}E[\zeta'] \\ &= \frac{2^{b'}}{2^{b'}-1}\left(y-\frac{2^{-\sigma_1(y+z_{\min})}}{2^{b'}-1}E[\zeta']\right) \end{aligned}\tag{4.86}$$

Finally, the results in (4.77) and (4.86) help us bound (4.65) as follows:

$$\begin{aligned}&\sum_{j=1}^{\infty}2^{b'j}\sum_{\mu\geq y}\left(\frac{\mu-y}{\Delta}+1\right)f_j(y,\mu) \\ =\ &\sum_{j=1}^{\infty}2^{b'j}\sum_{\mu\geq y}f_j(y,\mu)+\Delta^{-1}\sum_{j=1}^{\infty}2^{b'j}\sum_{\mu\geq y}(\mu-y)f_j(y,\mu) \\ \leq\ &\frac{2^{-\sigma_1(y+z_{\min})-b'}-1}{1-2^{-b'}}+\frac{\Delta^{-1}}{1-2^{-b'}}\left(y-\frac{2^{-\sigma_1(y+z_{\min})}}{2^{b'}-1}E[\zeta']\right) \\ =\ &\frac{2^{-\sigma_1(y+z_{\min})-b'}-1}{1-2^{-b'}}+\frac{\Delta^{-1}}{1-2^{-b'}}\left(y-\frac{2^{-\sigma_1(y+z_{\min})-b'}}{1-2^{-b'}}E[\zeta']\right) \\ =\ &\left(\frac{2^{-b'}}{1-2^{-b'}}-\frac{\Delta^{-1}2^{-b'}E[\zeta']}{(1-2^{-b'})^2}\right)2^{-\sigma_1(y+z_{\min})}+\frac{\Delta^{-1}y-1}{1-2^{-b'}} \\ =\ &C_1'2^{-\sigma_1 y}-\frac{1-\Delta^{-1}y}{1-2^{-b'}}\end{aligned}\tag{4.87}$$

where

$$C_1'=\frac{2^{-b'}}{1-2^{-b'}}-\frac{\Delta^{-1}2^{-b'}E[\zeta']}{(1-2^{-b'})^2}$$

Clearly C_1' is a positive quantity since the expected branch metric along a wrong path, or $E[\zeta']$, is negative. By applying (4.87) in (4.65), we are finally able to bound the average number of computations performed in τ_0' as follows:

$$\begin{aligned}&\sum_{j=1}^{\infty}E\left[\sum_{\boldsymbol{v}_j\in\tau_0'}\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\middle|T_{\min}^*=y\right] \\ \leq\ &(1-2^{-b})C_1'2^{-\sigma_1 y}-\frac{1-2^{-b}}{1-2^{-b'}}\left(-\frac{y}{\Delta}+1\right)\end{aligned}\tag{4.88}$$

4.2.5 On partially true paths

To repeat the previous analysis for the paths in τ_0^*, we again begin with the base case for nodes at depth 1. We can infer from Fig. 4.1 that besides the node $\boldsymbol{v}_1^*$ on the correct path, there are $(i_{\max}+d_{\max})$ other nodes that also point to the true encoder state s_1^*, and thus predict the transmitted block accurately. Hence unlike the previous case, we cannot assume statistical independence between the predicted block output and the received frame, implying that the branch metrics in τ_0^* follow a different probability distribution. Hence, we let a random variable $Z^*(\delta)$ characterize the distribution of metrics along such branches, for a specific drift change δ. Now to bound the number of visits to all nodes in τ_0^* at depth 1, we write:

$$\begin{aligned} & E\Big[\sum_{\boldsymbol{v}_1\in\tau_0^*}\Big(\frac{\mu(\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)\Big|T_{\min}^*=y\Big] \\ = & \sum_{\delta_1^*}P(\delta_1^*)\sum_{\boldsymbol{v}_1\in\tau_0^*}E\Big[\Big(\frac{\mu(\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)|T_{\min}^*=y,\delta_1^*\Big] \\ = & \sum_{\delta_1^*}P(\delta_1^*)\sum_{\substack{\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z^*(\delta_1)=z_1) \end{aligned} \tag{4.89}$$

To extend this equation for the analysis of longer paths in τ_0^*, we first acknowledge that such paths may wholly or partially indicate the true transmitted frame, but suggest an alternate sequence of drift states. It is also worth noting that there may be multiple paths with alternate drift sequences that yield the same aposteriori probability as the correct path p^*. For instance, an all-zero codeword transmitted with some non-zero final drift, could be described by multiple drift sequences that appear equally likely. Here, we choose to limit our attention to more well-behaved codewords, wherein such an uncertainty in drift values is much lower. Hence, most of the paths in τ_0^* that pertain to the correct message sequence, correspond to drift sequences which are implausible. Beyond a certain depth, the majority of these paths accumulate a significant deviation from the true drift sequence as predicted by p^*. The consequent shift between the received and hypothesized sequences makes them appear random with respect to each other, despite the associated message sequences being identical. In light of this, we can reasonably assume that even for any $\boldsymbol{v}_j \in \tau_0^*$ that predicts the transmitted sequence accurately, the output block hypothesized by the branch $\boldsymbol{v}_{j-1} \to \boldsymbol{v}_j$ is statistically independent of the corresponding received block, if this branch lies at a sufficient depth in the code tree.
For analytical convenience, we consider this assumption to hold from depth 2. Hence for any $\boldsymbol{v}_j \in \tau_0^*$ with the associated sequence of drift changes being $\delta_1, \cdots \delta_j$, the probability

distribution of branch metrics is considered to be:

$$P(Z_1^*(\delta_1) = z_1, Z_2'(\delta_2) = z_2, \cdots Z_j'(\delta_j) = z_j) = P(Z_1^*(\delta_1) = z_1) \prod_{i=2}^{j} P(Z_i'(\delta_i) = z_i) \quad (4.90)$$

Here, the equality follows from the assumption of independence of branch metrics along an incorrect path. With such a characterization of the probability distribution of branch metrics along a path, we can proceed to generalize (4.89):

$$\begin{aligned} & \sum_{\boldsymbol{v}_j \in \tau_0^*} E\left[\left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1\right)\phi(\boldsymbol{v}_j)\middle| T_{\min}^* = y\right] \\ = \; & 2^{b(j-1)} \sum_{\delta_1^* = -d_{\max}}^{i_{\max}} P(\delta_1^*) \sum_{\substack{\boldsymbol{\delta}_1^j \\ \delta_1 \neq \delta_1^*}} \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1\right) P(Z_1^*(\delta_1) = z_1) \prod_{i=2}^{j} P(Z_i'(\delta_i) = z_i) \end{aligned} \quad (4.91)$$

To simplify this expression, we may once again use (4.59) to do away with the drift-specific distributions for metrics of incorrect branches:

$$\begin{aligned} & \sum_{\substack{\boldsymbol{\delta}_1^j \\ \delta_1 \neq \delta_1^*}} \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1\right) P(Z_1^*(\delta_1) = z_1) \prod_{i=2}^{j} P(Z_i'(\delta_i) = z_i) \\ = \; & \sum_{\substack{\delta_1 = -d_{\max} \\ \delta_1 \neq \delta_1^*}}^{i_{\max}} \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} P(Z_1^*(\delta_1) = z_1) \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1\right) \sum_{(\delta_2, \cdots \delta_j)} \prod_{i=2}^{j} P(Z_i'(\delta_i) = z_i) \\ = \; & \lambda^{j-1} \sum_{\substack{\delta_1 = -d_{\max} \\ \delta_1 \neq \delta_1^*}}^{i_{\max}} \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} P(Z_1^*(\delta_1) = z_1) \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1\right) \prod_{i=2}^{j} P(\zeta_i' = z_i) \end{aligned} \quad (4.92)$$

In a manner similar to (4.62), we can redefine all paths in terms of their respective node metrics so as to further simplify the above equation:

$$\begin{aligned} & \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} P(Z_1^*(\delta_1) = z_1) \left(\frac{\sum_{i=1}^{j} z_i - y}{\Delta} + 1\right) \prod_{i=2}^{j} P(\zeta_i' = z_i) \\ = \; & \sum_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^{(j)}(y)} P(Z_1^*(\delta_1) = z_1) \left(\frac{\sum_{i=2}^{j} z_i - (y - z_1)}{\Delta} + 1\right) \prod_{i=2}^{j} P(\zeta_i' = z_i) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\substack{(z_2,\cdots z_j) \\ \in \boldsymbol{\mathcal{Z}}^{(j-1)}(y-z_1)}} \left(\frac{\sum_{i=2}^{j} z_i - (y - z_1)}{\Delta} + 1 \right) \prod_{i=2}^{j} P(\zeta_i' = z_i) \\
&= \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\substack{(\mu_2,\cdots \mu_j) \\ \in \boldsymbol{\mathcal{M}}^{(j-1)}(y-z_1)}} \left(\frac{\mu_j - (y - z_1)}{\Delta} + 1 \right) P(M_2' = \mu_2, \cdots, M_j' = \mu_j) \\
&= \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\mu_j \geq y - z_1} \left(\frac{\mu_j - (y - z_1)}{\Delta} + 1 \right) f_{j-1}(y - z_1, \mu_j) \qquad (4.93)
\end{aligned}
$$

Upon incorporating (4.92) and (4.93) in (4.91), we arrive at:

$$
\begin{aligned}
&\sum_{\boldsymbol{v}_j \in \tau_0^*} E\left[\left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_j) \middle| T_{\min}^* = y \right] \\
&= (2^b \lambda)^{j-1} \sum_{\substack{\delta_1^*, \delta_1 = -d_{\max} \\ \delta_1 \neq \delta_1^*}}^{i_{\max}} P(\delta_1^*) \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\mu \geq y - z_1} \left(\frac{\mu - (y - z_1)}{\Delta} + 1 \right) f_{j-1}(y - z_1, \mu) \\
&= 2^{b'(j-1)} \sum_{\substack{\delta_1^*, \delta_1 = -d_{\max} \\ \delta_1 \neq \delta_1^*}}^{i_{\max}} P(\delta_1^*) \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\mu \geq y - z_1} \left(\frac{\mu - (y - z_1)}{\Delta} + 1 \right) f_{j-1}(y - z_1, \mu)
\end{aligned}
$$

We may also rewrite this equation as:

$$
\begin{aligned}
&\sum_{\boldsymbol{v}_{j+1} \in \tau_0^*} E\left[\left(\frac{\mu(\boldsymbol{v}_{j+1}) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_{j+1}) \middle| T_{\min}^* = y \right] \\
&= 2^{b'j} \sum_{\substack{\delta_1^*, \delta_1 = -d_{\max} \\ \delta_1 \neq \delta_1^*}}^{i_{\max}} P(\delta_1^*) \sum_{z_1 \geq y} P(Z_1^*(\delta_1) = z_1) \sum_{\mu \geq y - z_1} \left(\frac{\mu - (y - z_1)}{\Delta} + 1 \right) f_j(y - z_1, \mu)
\end{aligned} \qquad (4.94)
$$

Now to bound the average number of computations performed in τ_0^* for a specific minimum threshold as required in (4.53), we may write that:

$$
\begin{aligned}
&\sum_{j=1}^{\infty} E\left[\sum_{\boldsymbol{v}_j \in \tau_0^*} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_j) \middle| T_{\min}^* = y \right] \\
&= E\left[\sum_{\boldsymbol{v}_1 \in \tau_0^*} \left(\frac{\mu(\boldsymbol{v}_1) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_1) \middle| T_{\min}^* = y \right] \\
&\quad + \sum_{j=2}^{\infty} E\left[\sum_{\boldsymbol{v}_j \in \tau_0^*} \left(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1 \right) \phi(\boldsymbol{v}_j) \middle| T_{\min}^* = y \right]
\end{aligned}
$$

$$
\begin{aligned}
&= E\Big[\sum_{\boldsymbol{v}_1\in\tau_0^*}\Big(\frac{\mu(\boldsymbol{v}_1)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_1)\Big|T^*_{\min}=y\Big] \\
&\quad+\sum_{j=1}^{\infty}E\Big[\sum_{\boldsymbol{v}_{j+1}\in\tau_0^*}\Big(\frac{\mu(\boldsymbol{v}_{j+1})-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_{j+1})\Big|T^*_{\min}=y\Big]
\end{aligned}
$$

With equations (4.89) and (4.94), we can further reduce the above expression as follows:

$$
\begin{aligned}
&\sum_{j=1}^{\infty}E\Big[\sum_{\boldsymbol{v}_j\in\tau_0^*}\Big(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_j)\Big|T^*_{\min}=y\Big] \\
=\;&\sum_{\delta_1^*}P(\delta_1^*)\sum_{\substack{\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z^*(\delta_1)=z_1) \\
&+\sum_{j=1}^{\infty}2^{b'j}\sum_{\substack{\delta_1^*,\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)\sum_{\mu\geq y-z_1}\Big(\frac{\mu-(y-z_1)}{\Delta}+1\Big)f_j(y-z_1,\mu) \\
=\;&\sum_{\delta_1^*}P(\delta_1^*)\sum_{\substack{\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z^*(\delta_1)=z_1) \\
&+\sum_{\substack{\delta_1^*,\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)\sum_{j=1}^{\infty}2^{b'j}\sum_{\mu\geq y-z_1}\Big(\frac{\mu-(y-z_1)}{\Delta}+1\Big)f_j(y-z_1,\mu)
\end{aligned}
\tag{4.95}
$$

By reusing the inequality specified in (4.87), we can proceed to further simplify the preceding equation to obtain:

$$
\begin{aligned}
&\sum_{j=1}^{\infty}E\Big[\sum_{\boldsymbol{v}_j\in\tau_0^*}\Big(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\Big)\phi(\boldsymbol{v}_j)\Big|T^*_{\min}=y\Big] \\
\leq\;&\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}P(\delta_1^*)\sum_{z_1\geq y}\Big(\frac{z_1-y}{\Delta}+1\Big)P(Z_1^*(\delta_1)=z_1) \\
&+C_1'\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{-\sigma_1(y-z_1)} \\
&-\frac{1}{1-2^{-b'}}\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}}P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)\Big(1-\Delta^{-1}(y-z_1)\Big)
\end{aligned}
$$

$$
\begin{aligned}
&= \left(1-\frac{1}{1-2^{-b'}}\right)\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}\left(\frac{z_1-y}{\Delta}+1\right)P(Z_1^*(\delta_1)=z_1)\\
&\quad +C_1'\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{\sigma_1(z_1-y)}\\
&= -\frac{2^{-b'}}{1-2^{-b'}}\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}\left(\frac{z_1-y}{\Delta}+1\right)P(Z_1^*(\delta_1)=z_1)\\
&\quad +C_1'\sum_{\substack{\delta_1^*,\delta=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{\sigma_1(z_1-y)}\\
&\leq C_1'\sum_{\substack{\delta_1^*,\delta_1=-d_{\max}\\ \delta_1\neq\delta_1^*}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{\sigma_1(z_1-y)}\\
&= C_1'2^{-\sigma_1 y}\sum_{\delta_1^*,\delta_1=-d_{\max}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{\sigma_1 z_1}\\
&\quad -C_1'2^{-\sigma_1 y}\sum_{\delta_1^*=-d_{\max}}^{i_{\max}} P(\delta_1^*)\sum_{z_1\geq y}P(Z_1^*(\delta_1^*)=z_1)2^{\sigma_1 z_1} \qquad (4.96)
\end{aligned}
$$

To simplify the latter term in (4.96), we introduce another mixture distribution, analogous to (4.55):

$$P(\zeta^*=z)=\sum_{\delta=-d_{\max}}^{i_{\max}} P(\delta)P(Z^*(\delta)=z) \qquad (4.97)$$

This essentially represents the probability distribution of branch metrics along the correct path, since $Z^*(\delta)$ accounts for the errors during transmission for a specific drift change δ, while $P(\delta)$ appropriately weights these drift-specific distributions according to the likelihood of net insertions or deletions over a single block. Hence, (4.96) becomes:

$$
\begin{aligned}
&\sum_{j=1}^{\infty} E\Bigg[\sum_{\boldsymbol{v}_j\in\tau_0^*}\left(\frac{\mu(\boldsymbol{v}_j)-y}{\Delta}+1\right)\phi(\boldsymbol{v}_j)\Bigg|T_{\min}^*=y\Bigg]\\
&\leq C_1'2^{-\sigma_1 y}\left(\sum_{\delta_1=-d_{\max}}^{i_{\max}}\sum_{z_1\geq y}P(Z_1^*(\delta_1)=z_1)2^{\sigma_1 z_1}-\sum_{z_1\geq y}P(\zeta_1^*=z_1)2^{\sigma_1 z_1}\right)\\
&\leq C_1'2^{-\sigma_1 y}\left(\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)-\sum_{z_1\geq y}P(\zeta_1^*=z_1)2^{\sigma_1 z_1}\right) \qquad (4.98)
\end{aligned}
$$

where $g_{0,\delta}(\sigma)$ represents a moment generating function of the drift-specific distribution of branch metrics along a correct path, $P(Z^*(\delta) = z)$. Evidently for a specific BSID channel, $\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)$ is a constant.
Now to deal with the second term in (4.98), we make use of a value $z^*_{\min}$, which refers to the minimum value that ζ^* can assume. Hence,

$$\begin{aligned}\sum_{z_1 \geq y} P(\zeta_1^* = z_1) 2^{\sigma_1 z_1} &\geq 2^{\sigma_1 z^*_{\min}} \sum_{z_1 \geq y} P(\zeta_1^* = z_1) \\ &= 2^{\sigma_1 z^*_{\min}} P(\zeta^* \geq y) \\ &= 2^{\sigma_1 z^*_{\min}} P(2^{\sigma_0 \zeta^*} \leq 2^{\sigma_0 y})\end{aligned}$$

where σ_0 as initially introduced in (4.11), is negative and satisfies $g_0(\sigma_0) = 1$.
In a similar spirit as (4.11), the above inequality can be further reduced as follows:

$$\begin{aligned}\sum_{z_1 \geq y} P(\zeta_1^* = z_1) 2^{\sigma_1 z_1} &\geq 2^{\sigma_1 z^*_{\min}} (1 - P(2^{\sigma_0 \zeta^*} > 2^{\sigma_0 y})) \\ &\geq 2^{\sigma_1 z^*_{\min}} (1 - 2^{-\sigma_0 y} g_0(\sigma_0)) \\ &= 2^{\sigma_1 z^*_{\min}} (1 - 2^{-\sigma_0 y})\end{aligned}$$

By incorporating this in (4.98), we finally arrive at the following bound on the average number of computations due to examination of paths in the subset τ_0^*, for a given value of minimum threshold:

$$\begin{aligned}&\sum_{j=1}^{\infty} E\Big[\sum_{\boldsymbol{v}_j \in \tau_0^*} \Big(\frac{\mu(\boldsymbol{v}_j) - y}{\Delta} + 1 \Big) \phi(\boldsymbol{v}_j) \Big| T^*_{\min} = y \Big] \\ \leq\ & C_1' 2^{-\sigma_1 y} \Big(\sum_{\delta_1 = -d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1) - 2^{\sigma_1 z^*_{\min}} (1 - 2^{-\sigma_0 y}) \Big) \\ =\ & \Big(\sum_{\delta_1 = -d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1) - 2^{\sigma_1 z^*_{\min}} \Big) C_1' 2^{-\sigma_1 y} + C_1' 2^{\sigma_1 z^*_{\min} - (\sigma_1 + \sigma_0) y} \qquad (4.99)\end{aligned}$$

4.2.6 Average decoding effort per block

Thus far, we have managed to bound the average decoding effort expended by a sequential decoder in the path subsets τ_0' and τ_0^* for a given value of minimum threshold $T^*_{\min}$. These results can be found in the inequalities (4.88) and (4.99) respectively. When put together, they lead us to a bound on the average number of computations performed in the entire incorrect subtree, τ_0.
By additionally including the number of visits to the root node, as in (4.52), we now

hope to determine the average number of computations necessary to accurately decode the first block $\boldsymbol{v}_0^*$, for a specific $T_{\min}^*$:

$$\begin{aligned}
E[C(\boldsymbol{v}_0^*)|T_{\min}^*=y] &\leq -\frac{y}{\Delta}+1+\big(\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)-2^{\sigma_1 z_{\min}^*}\big)C_1'2^{-\sigma_1 y} \\
&\quad +C_1'2^{\sigma_1 z_{\min}^*-(\sigma_1+\sigma_0)y}+(1-2^{-b})C_1'2^{-\sigma_1 y}-\frac{1-2^{-b}}{1-2^{-b'}}\Big(-\frac{y}{\Delta}+1\Big) \\
&= \big(1-2^{-b}+\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)-2^{\sigma_1 z_{\min}^*}\big)C_1'2^{-\sigma_1 y} \\
&\quad +C_1'2^{\sigma_1 z_{\min}^*-(\sigma_0+\sigma_1)y}+\Big(1-\frac{1-2^{-b}}{1-2^{-b'}}\Big)\Big(-\frac{y}{\Delta}+1\Big) \\
&= \big(1-2^{-b}+\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)-2^{\sigma_1 z_{\min}^*}\big)C_1'2^{-\sigma_1 y} \\
&\quad +C_1'2^{\sigma_1 z_{\min}^*-(\sigma_0+\sigma_1)y}+\frac{2^{-b}-2^{-b'}}{1-2^{-b'}}\Big(-\frac{y}{\Delta}+1\Big) \\
&= C_1 2^{-\sigma_1 y}+C_2 2^{-(\sigma_0+\sigma_1)y}+C_3\Big(-\frac{y}{\Delta}+1\Big) \qquad (4.100)
\end{aligned}$$

where

$$\begin{aligned}
C_1 &= \big(1-2^{-b}+\sum_{\delta_1=-d_{\max}}^{i_{\max}} g_{0,\delta_1}(\sigma_1)-2^{\sigma_1 z_{\min}^*}\big)C_1' \\
C_2 &= C_1'2^{\sigma_1 z_{\min}^*} \\
C_3 &= \frac{2^{-b}-2^{-b'}}{1-2^{-b'}}
\end{aligned}$$

Now for the next step, thus involves averaging this quantity over all possible values of $T_{\min}^*$ as in (4.8). In this manner, we can bound the average decoding complexity per block:

$$\begin{aligned}
C_{\text{av}} &= \sum_{i=0}^{\infty} P(T_{\min}^*=-i\Delta)E[C(\boldsymbol{v}_0^*)|T_{\min}^*=-i\Delta] \\
&= \sum_{i=0}^{\infty} P(T_{\min}^*=-i\Delta)\big(C_1 2^{\sigma_1 i\Delta}+C_2 2^{(\sigma_0+\sigma_1)i\Delta}+C_3(i+1)\big) \\
&= C_1\sum_{i=0}^{\infty} P(T_{\min}^*=-i\Delta)2^{\sigma_1 i\Delta}+C_2\sum_{i=0}^{\infty} P(T_{\min}^*=-i\Delta)2^{(\sigma_0+\sigma_1)i\Delta}
\end{aligned}$$

$$+C_3 \sum_{i=0}^{\infty} P(T^*_{\min} = -i\Delta)(i+1) \tag{4.101}$$

The individual summations in (4.101) can be dealt with in the following manner:

$$\begin{aligned}
\sum_{i=0}^{\infty} P(T^*_{\min} = -i\Delta)2^{\sigma_1 i\Delta} &= \sum_{i=0}^{\infty}(P(T^*_{\min} \leq -i\Delta) - P(T^*_{\min} \leq -(i+1)\Delta))2^{\sigma_1 i\Delta} \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 i\Delta} - \sum_{i=0}^{\infty} P(T^*_{\min} \leq -(i+1)\Delta)2^{\sigma_1 i\Delta} \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 i\Delta} - \sum_{i=1}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 (i-1)\Delta} \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 i\Delta} - \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 (i-1)\Delta} \\
&\quad +2^{-\sigma_1\Delta}P(T^*_{\min} \leq 0) \\
&= (1-2^{-\sigma_1\Delta})\sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{\sigma_1 i\Delta} + 2^{-\sigma_1\Delta}
\end{aligned} \tag{4.102}$$

The final equality follows from the fact that $T^*_{\min}$ can never assume a positive value. Similar to (4.102), we may also write:

$$\begin{aligned}
\sum_{i=0}^{\infty} P(T^*_{\min} = -i\Delta)2^{(\sigma_0+\sigma_1)i\Delta} &= (1-2^{-(\sigma_1+\sigma_0)\Delta})\sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)2^{(\sigma_1+\sigma_0)i\Delta} \\
&\quad +2^{-(\sigma_1+\sigma_0)\Delta}
\end{aligned} \tag{4.103}$$

To deal with the final summation in (4.101), we undertake a similar approach:

$$\begin{aligned}
\sum_{i=0}^{\infty} P(T^*_{\min} = -i\Delta)(i+1) &= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta) - P(T^*_{\min} \leq -(i+1)\Delta)(i+1) \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)(i+1) - \sum_{i=1}^{\infty} P(T^*_{\min} \leq -i\Delta)i \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)(i+1) - \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)i \\
&= \sum_{i=0}^{\infty} P(T^*_{\min} \leq -i\Delta)
\end{aligned} \tag{4.104}$$

By incorporating (4.102), (4.103) and (4.104) in (4.101), and additionally exploiting the bound on minimum threshold as stated in (4.13), we can further simplify our bound on

average decoding complexity per block:

$$\begin{aligned} C_{\text{av}} &\leq C_1(1-2^{-\sigma_1\Delta})\sum_{i=0}^{\infty}2^{\sigma_1 i\Delta}2^{\sigma_0(i-1)\Delta}+C_1 2^{-\sigma_1\Delta}+C_3\sum_{i=0}^{\infty}2^{\sigma_0(i-1)\Delta} \\ &\quad +C_2(1-2^{-(\sigma_0+\sigma_1)\Delta})\sum_{i=0}^{\infty}2^{(\sigma_1+\sigma_0)i\Delta}2^{\sigma_0(i-1)\Delta}+C_2 2^{-(\sigma_0+\sigma_1)\Delta} \\ &= C_1(1-2^{-\sigma_1\Delta})2^{-\sigma_0\Delta}\sum_{i=0}^{\infty}2^{(\sigma_1+\sigma_0)i\Delta}+C_3\sum_{i=0}^{\infty}2^{\sigma_0(i-1)\Delta} \\ &\quad +C_2(1-2^{-(\sigma_0+\sigma_1)\Delta})2^{-\sigma_0\Delta}\sum_{i=0}^{\infty}2^{(\sigma_1+2\sigma_0)i\Delta}+C_1 2^{-\sigma_1\Delta}+C_2 2^{-(\sigma_0+\sigma_1)\Delta} \end{aligned}$$

Since σ_0 is already known to be negative, the sufficient condition for the convergence of this upper bound is:

$$\sigma_0+\sigma_1<0 \tag{4.105}$$

If these criteria are met, the average decoding effort per block can be bounded by:

$$\begin{aligned} C_{\text{av}} &\leq C_1\frac{(1-2^{-\sigma_1\Delta})2^{-\sigma_0\Delta}}{1-2^{(\sigma_0+\sigma_1)\Delta}}+C_3\frac{2^{-\sigma_0\Delta}}{1-2^{\sigma_0\Delta}} \\ &\quad +C_2\frac{(1-2^{-(\sigma_0+\sigma_1)\Delta})2^{-\sigma_0\Delta}}{1-2^{(\sigma_1+2\sigma_0)\Delta}}+C_1 2^{-\sigma_1\Delta}+C_2 2^{-(\sigma_0+\sigma_1)\Delta} \end{aligned} \tag{4.106}$$

5 Numerical Results

In this chapter, we aim to evaluate two major aspects of Fano's decoder, namely its decoding accuracy and computational complexity. To this end, we compare its performance in these respects with the Viterbi decoder, and the convolutional codes chosen for this comparison are outlined in Table 5.1. Here, the generator polynomials are stated in octal form.

We confine our attention to BSID channels with parameters set to $P_i = P_d$ and $P_s = 0$. Additionally in order to limit complexity of both decoders, we restrict the maximum number of insertions and deletions considered over a single block of c bits to $i_{\max} = d_{\max} = c$. Besides this, drift states with magnitudes beyond $\hat{d}_{\max} = 30$ are ignored.

Code	(c, b)	m	$\boldsymbol{g}$			d_{free}
C1	$(3, 1)$	1	1	3	3	5
C2	$(3, 1)$	6	117	127	155	14

Table 5.1: Convolutional codes for testing

5.1 Frame Error Rate

In order to examine the decoding performance of Fano's sequential decoder, we consider terminated codewords that encode $L = 100$ information blocks, and conduct simulations for the aforementioned BSID channel. This process is also repeated for the Viterbi decoder and the resulting frame error rates are illustrated in Fig. 5.1. We find that Fano's decoder is clearly outperformed by the Viterbi decoder for both C1 and C2. This actually falls in line with our theoretical intuition since a sequential decoder, as highlighted earlier in Chapter 3, is sub-optimal due to its partial examination of a code tree. On the other hand, a Viterbi decoder which accounts for all possible trellis paths and thus provides a maximum-likelihood estimate.

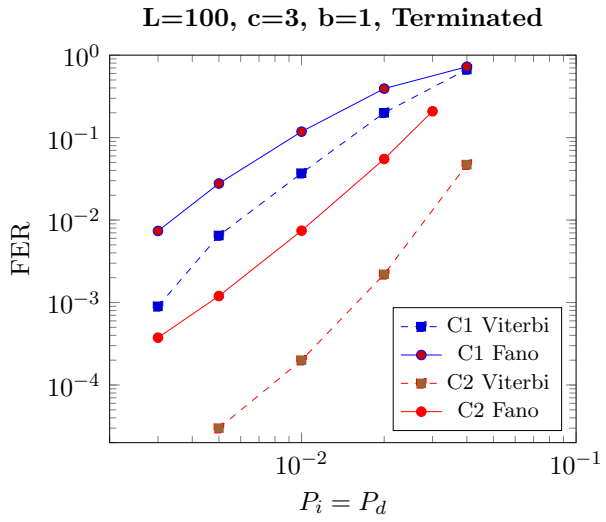


Figure 5.1: Frame error rate

5.2 Computational Cutoff Rate

The theoretical bound for average decoding effort required per block, as stated in (4.106), is found to be several orders of magnitude higher than its actual value. This discrepancy can attributed to several reasons. Firstly, the complexity analysis is conducted for a random ensemble of codes instead of the specific codes used for simulation. In [**?**], a comparison of both cases revealed that for binary symmetric channels, a decline in metric along the correct path was more likely to occur for random codes, thereby implying more decoding effort. Furthermore, the assumption of infinite tree length does not hold, since neither code in Table 5.1 has infinite memory. This in turn, also invalidates the idea that the average decoding effort required per block is identically distributed for all blocks in a codeword.

However, this bound is still significant as the conditions necessary for its existence lead us to the computational cutoff rate R_0. Effectively, this is the rate beyond which the use of a sequential decoder becomes computationally impractical. This is due to the fact that for rates exceeding R_0, the channel noise is relatively severe and causes the decoder to backtrack quite frequently. Thus, as the code tree becomes longer due to the transmission of more information blocks, the number of computations performed in incorrect subtrees increases quite rapidly. For trees of infinite length, the average decoding effort required per block or C_{av}, would be unbounded, and this corresponds precisely to the case when condition (4.105) is not met. On the contrary, for code rates lower than the cutoff rate, C_{av} is bounded and hence, the number of computations needed to decode a single frame is expected to grow linearly with the number of information blocks transmitted.

The preceding discussion suggests that determination of the cutoff rate R_0 simply involves finding the rate at which the following criterion holds:

$$\sigma_0 + \sigma_1 = 0 \tag{5.1}$$

Incidentally, this makes way for a simple intuitive interpretation of R_0. We observe from (4.12) that probability of the minimum node metric along the correct path dipping below a value y, decreases exponentially with y. At the same time, (4.78) and (4.98) suggest that the number of nodes in τ_0 that stay above the corresponding minimum threshold and hence may be visited by the decoder, increases exponentially with y. Hence, when condition (4.105) is satisfied, the rate of decline of probability $P(\mu^*_{\min} < y)$ or σ_0, sufficiently compensates the rate at which the number of wasteful computations increase, thereby allowing the convergence of C_{av}.

From the preceding discussion, it is easy to see that as channel noise becomes more severe, higher redundancy is necessary to ensure that node metrics along the correct path do not dip too sharply. This is exactly what we observe in Fig. 5.2. Since the values of σ_0 and σ_1 depend solely on the distribution of branch metrics along a correct or incorrect path, the apparent independence of R_0 from encoder memory is also quite expected. However, one strange aspect of its behavior is that the cutoff rate seems to depend on the specific values of b and c, instead of just their ratio. This peculiarity seems to fade as these code parameters assume higher values, and can perhaps be explained by the path merging phenomenon described in Section 3.3. To put it more explicitly, since branches representing longer blocks combine multiple paths in the corresponding code tree for shorter blocks, their metrics follow a distribution that cannot be defined in terms of the branch metrics for shorter blocks. Consequently, behavior of the roots σ_0 and σ_1 differ for distinct tuples of (c, b) that denote the same rate, thereby causing the cutoff rate to also act similarly.

5.3 Simulated Complexity

Now to compare the practical complexity of the two algorithms, we first need to recognize that for any decoder, the net computational effort is essentially dependent on the total number of branch metric computations performed.

Hence, to determine the complexity of decoding a specific received frame by means of a Viterbi decoder, we simply need to ascertain the total number of branches in the associated trellis. To proceed along these lines, it suffices to limit our focus on the initial divergent part of a trellis, as the memory of either code in Table 5.1 is far lower than the

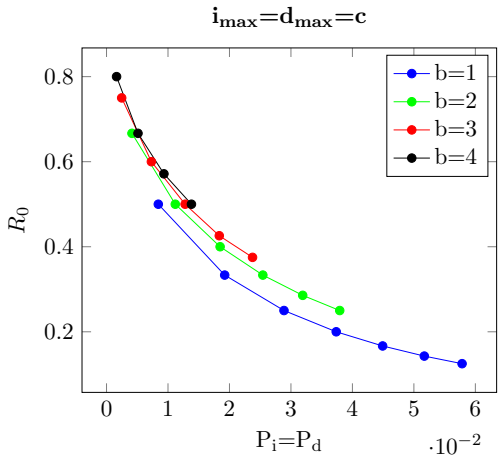


Figure 5.2: Computational cutoff rate

number of information blocks being considered. Now in this portion of the trellis, each node gives rise to $2^b(i_{\max} + d_{\max} + 1)$ outgoing branches. Additionally, the number of nodes at a certain depth j, may be seen as a product of the number of encoder states and drift states at that depth. Thus, the total number of branches in the initial part of a trellis can be expressed as:

$$\sum_{i=0}^{L-1} \mathcal{N}_j \mathcal{P}_j 2^b (i_{\max} + d_{\max} + 1) \tag{5.2}$$

where $\mathcal{N}_j$ and $\mathcal{P}_j$ denote the number of encoder and drift states in a trellis at depth j respectively. They can be defined as follows:

$$\mathcal{N}_j = \begin{cases} 2^j, & \text{if } j \leq m \\ 2^m, & \text{else} \end{cases} \tag{5.3}$$

$$\mathcal{P}_j = \min(ji_{\max} + jd_{\max} + 1, 2\hat{d}_{\max} + 1) \tag{5.4}$$

As for Fano's algorithm, we have already seen in Chapter 4 that the complexity of decoding a certain frame is variable, and depends on factors like the code being used, the relative locations of the errors, and so on. Thus, we resort to simply determining the average decoding complexity for each set of channel parameters. This is accomplished by recognizing that every time the decoder steps forward to visit a node, it evaluates the metrics of each of the $2^b(i_{\max} + d_{\max} + 1)$ branches that emerge from it. Consequently,

the average number of computations required to decode a single frame is given by:

$$\mathcal{F}_{\text{av}} 2^b (i_{\max} + d_{\max} + 1) \tag{5.5}$$

where $\mathcal{F}_{\text{av}}$ refers to the average number of forward steps taken by the decoder. Using (5.2) and (5.5), we arrive at the following complexity reduction factor:

$$\nu = \frac{\sum_{i=0}^{L-1} \mathcal{N}_j \mathcal{P}_j}{\mathcal{F}_{\text{av}}} \tag{5.6}$$

The variation of this quantity with increasing insertion and deletion probabilities has been depicted in Fig. 5.3. Additionally using 5.1, we find that probability P^*, as marked in Fig. 5.3, corresponds to channel operation at the computational cutoff rate R_0.
It is observed that as channel noise becomes more severe, especially beyond P^*, sequential decoding gradually loses its computational advantage to Viterbi's algorithm. This is due to the fact that as error probabilities increase, the node metrics along the correct path are more likely to exhibit frequent dips in value, similar to Eg. 3.5.1. As a result, the decoder is forced to examine many more incorrect paths before the correct one is found.
To study the impact of the path merging phenomenon, we repeat these simulations for two new codes C1' and C2', which are essentially identical to C1 and C2, but with parameters $c = 6$ and $b = 2$. The results are illustrated in Fig. 5.4. As before, the probability $P^{*'}$ is meant to correspond to cutoff rate operation.

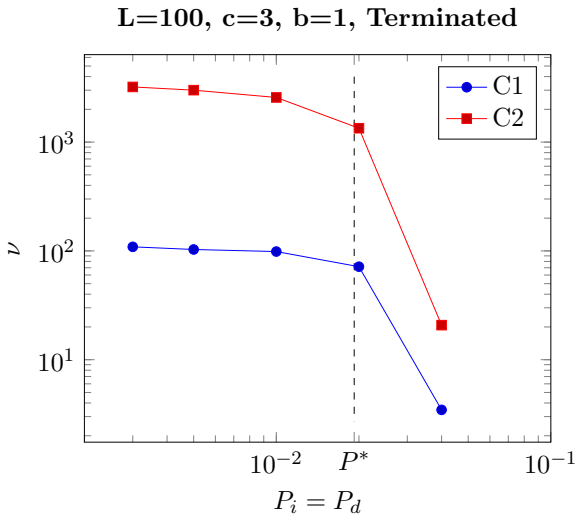


Figure 5.3: Computational complexity of $(3, 1)$ codes

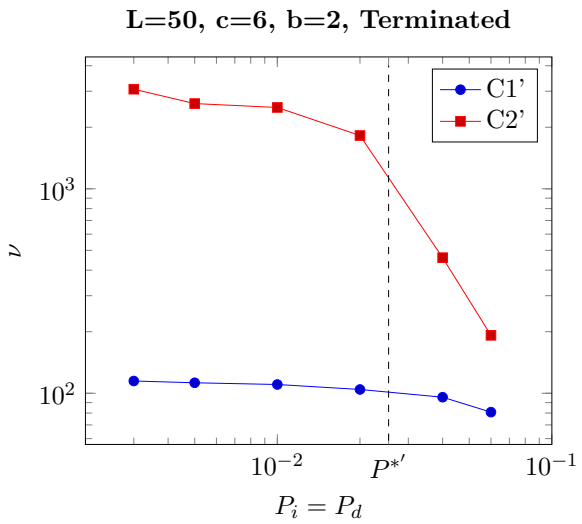


Figure 5.4: Computational complexity of $(6, 2)$ codes

6 Conclusion

The primary objective of this work was to expand the application of sequential decoders to channels that are susceptible to insertion, deletion and substitution errors. Particularly, we chose to limit our focus to Fano's algorithm. In order to facilitate the implementation of its decoder, a suitable decoding metric was also derived. Following this, we investigated its efficacy as a decoder for convolutional codes. Viterbi's algorithm was used as a benchmark.

Firstly, in regard to decoding accuracy, simulation results revealed that Fano's decoder is always outperformed by Viterbi's algorithm. This validated our theoretical intuitions, since Fano's decoder only examines codewords that appear to be promising, thereby potentially ignoring the correct one.

Subsequently, we considered the aspect of computational complexity. We expected Fano's decoder to offer significant reductions in decoding complexity, upon comparison with Viterbi's algorithm. This was found to be true, but only when noise levels were relatively low. As the channel's error probabilities worsened, the decoder seemed to step back as often as it moved forward, and thus gradually lost its computational advantage.

An effort was also made to study the complexity of Fano's decoder analytically. For simplicity, we considered a random ensemble of convolutional codes with infinite memory. An asymptotic expression of the decoding metric was obtained and thereafter, we characterized its distribution along correct and incorrect paths in the code tree. Eventually, an upper bound on the average decoding complexity per block is established. When compared with practical values, this bound is found to be several orders of magnitude higher and hence, does not appear to offer anything useful. However, the required criterion for its existence provides us a method of determining the computational cutoff rate. For a specific channel, it essentially refers to the code rate beyond which the use a sequential decoder becomes impractical, from the standpoint of complexity. This can be attributed to the issue of frequent backtracking under severe channel noise levels.

Future work could involve the investigation of alternate decoding metrics, or the error probability analysis of Fano's decoder. Furthermore, we can also explore random branching process techniques so as to obtain a legitimate bound on decoding complexity. Similar analyses may also be conducted for other variants of sequential decoders.