Recursive Decoding Procedures for Minimizing the Bit Error Probability for Block and Convolutional Codes

Vladimir B. Balakirsky

Euler Institute for Discrete Mathematics
and its Applications (EIDMA)
c/o Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
v.b.balakirsky@ele.tue.nl

Abstract

Recursive decoding procedures minimizing the bit error probability for block and convolutional codes and having complexities comparable to the complexity of maximum likelihood decoding are presented. The key idea of our approach to decoding problem is a representation of codewords as boundary points of the space consisting of rational–valued vectors whose components are probabilities. As a result, we arrive at the Viterbi–type algorithm, which constructs an optimal estimate of the transmitted information sequence, generalize considerations to the joint source–channel decoding, and present an algorithm for convolutional codes which does not require truncation of the code trellis.

1 Introduction

Consider a system of data transmission over a channel. Let some distortion function $d:\ \{0,1\}^k \times \{0,1\}^k \longrightarrow \mathsf{R}^+$ be assigned to each pair $(\mathbf{u},\hat{\mathbf{u}})$, where $\mathbf{u}$ and $\hat{\mathbf{u}}$ are transmitted and decoded information sequences, respectively. Suppose that the decoder has to minimize the average distortion under the given probability distribution on information sequences.

If d is the indicator function of the event $\mathbf{u} \neq \hat{\mathbf{u}}$ then the average distortion is known as the average decoding error probability, and it is minimized by the maximum *a posteriori* probability (MAP) decoder. To implement the decoding procedure one has to compute *a posteriori* probabilities for each information sequence and select a sequence with the largest value. Another well–known function is the Hamming distance between $\mathbf{u}$ and $\hat{\mathbf{u}}$. Then the average distortion is called "the bit error probability" and denoted by $P_{\rm b}$. The optimal decoding procedure minimizing $P_{\rm b}$ differs from the MAP decoding. It also requires computation of 2^k *a posteriori* probabilities, but the decision is based on values of 2^k sums, where each sum is the *a posteriori* distortion provided that a certain information sequence is decoded. A generalization of this procedure to convolutional codes leads to the

"BCJR algorithm" presented by the authors of [1] as the algorithm, "which is practical only for short codes". The problem of constructing effective algorithms for minimizing $P_{\rm b}$ attracted attention of many researches. The introductory part of [2] can be considered as a good overview of current understanding of the problem.

We address the following aspects of the problem. Let us consider the space $[0,1]^k$ consisting of all vectors of length k whose components belong to the interval $[0,1]$, and 2^k information sequences form the set of boundary points of this space. Given a received vector $\mathbf{y} \in \mathcal{Y}^n$, where $\mathcal{Y}$ is the output alphabet of the channel, the decoder recursively constructs the rational–valued vector $(\pi_1, \ldots, \pi_k) \in [0,1]^k$. We introduce the distance between this vector and the information sequence $\hat{\mathbf{u}} \in \{0,1\}^k$ in such a way that it is equal to the bit error probability if $\hat{\mathbf{u}}$ is decoded. This approach is then extended to codes generated by a convolutional encoder of memory ν and rate $1/c$ with the truncated code trellis. We describe the algorithm for constructing the vector $(\pi_1, \ldots, \pi_T) \in [0,1]^T$, where $T+\nu$ is the truncation level, in a similar way as the Viterbi decoding algorithm for minimizing the word error probability under the uniform probability distribution on information sequences. The only difference is the point that we exchange rational–valued vectors between buffers instead of binary vectors. It seems to be that this possibility leads to essential reduction of the complexity of the BCJR algorithm with the same final result. The approach mentioned above is also extended to the joint source–channel decoding when information sequences are formed by a source encoder with the use of a variable length prefix code. In particular, we show how to form an optimal estimate of the transmitted sequence of messages and avoid the situation when quantization of components of the result of decoding : set $\hat{u}_j = 0$ if $\pi_j \le 1/2$ and set $\hat{u}_j = 1$ if $\pi_j > 1/2$, for all $j = 1, \ldots, T$, leads to the sequence, which cannot be generated at the output of the source encoder. Finally, we present the algorithm for minimizing the bit error probability for convolutional codes that does not require truncation of the code trellis.

2 Recursive decoding procedure for block codes which does not use the structure of the code trellis

Let $Q(\mathbf{u}), \mathbf{u} = (u_1, \ldots, u_k) \in \{0,1\}^k$, be a given probability distribution on information sequences and let $\mathbf{x}(\mathbf{u}) = (x_1(\mathbf{u}), \ldots, x_n(\mathbf{u})) \in \mathcal{X}^n$ denote the codeword of an $|\mathcal{X}|$–ary block code of length n corresponding to the sequence $\mathbf{u}$. Let $\mathcal{Y}$ be a finite set and let the codeword $\mathbf{x}(\mathbf{u})$ be transmitted over a channel defined by conditional probabilities $W(\mathbf{y}|\mathbf{x}(\mathbf{u})), y \in \mathcal{Y}^n$, for all $\mathbf{u} \in \{0,1\}^k$. Denote

$$W(\mathbf{x}(\mathbf{u})|\mathbf{y}) \triangleq \frac{Q(\mathbf{u})W(\mathbf{y}|\mathbf{x}(\mathbf{u}))}{\sum_{\mathbf{u}'} Q(\mathbf{u}')W(\mathbf{y}|\mathbf{x}(\mathbf{u}'))}$$

Suppose that the decoder estimates the transmitted information sequence as the vector

$$\varphi(\mathbf{y}) = (\varphi_1(\mathbf{y}), \ldots, \varphi(\mathbf{y})) \in \{0,1\}^k$$

constructed in such a way that

$$Q(\varphi(\mathbf{y})) > 0$$

for all vectors $\mathbf{y}$ that can be received. Then *the bit error probability* can be expressed as

$$P_{\rm b} \triangleq \sum_{\mathbf{u}} \sum_{\mathbf{y}} Q(\mathbf{u})W(\mathbf{y}|\mathbf{x}(\mathbf{u}))\frac{d_{\rm H}(\mathbf{u}, \varphi(\mathbf{y}))}{k}$$

$$= \frac{1}{k}\sum_{\mathbf{y}}\Big[\sum_{\mathbf{u}'} Q(\mathbf{u}')W(\mathbf{y}|\mathbf{x}(\mathbf{u}'))\Big]\overline{d}_{\mathrm{H}}(\varphi(\mathbf{y})) \tag{1}$$

where d_{H} denotes the Hamming distance and

$$\overline{d}_{\mathrm{H}}(\varphi(\mathbf{y})) \triangleq \sum_{\mathbf{u}} W(\mathbf{x}(\mathbf{u})|\mathbf{y})d_{\mathrm{H}}(\mathbf{u},\varphi(\mathbf{y}))$$

Proposition. *Given* $\mathbf{y} \in \mathcal{Y}^n$, *let*

$$\pi_i(\mathbf{y}) \triangleq \sum_{\mathbf{u}:u_i=1} W(\mathbf{x}(\mathbf{u})|\mathbf{y}), \quad i = 1,\ldots,k.$$

Then

$$\overline{d}_{\mathrm{H}}(\varphi(\mathbf{y})) = \sum_{i=1}^{k} |\varphi_i(\mathbf{y}) - \pi_i(\mathbf{y})|.$$

If $Q(\hat{\mathbf{u}}) > 0$, *where* $\hat{\mathbf{u}} = (\hat{u}_1,\ldots,\hat{u}_k)$ *is constructed by the following component–wise mapping of the vector* $(\pi_1(\mathbf{y}),\ldots,\pi_k(\mathbf{y}))$:

$$\hat{u}_i = \begin{cases} 0, & \text{if } \pi_i(\mathbf{y}) \le 1/2 \\ 1, & \text{if } \pi_i(\mathbf{y}) > 1/2 \end{cases} \tag{2}$$

for all $i = 1,\ldots,k$, *then* $\varphi(\mathbf{y}) = \hat{\mathbf{u}}$ *minimizes the function* $\overline{d}_{\mathrm{H}}(\varphi(\mathbf{y}))$.

Proof :

$$\begin{aligned}
\sum_{i=1}^{k} |\varphi_i(\mathbf{y}) - \pi_i(\mathbf{y})| &= \sum_{i:\varphi_i(\mathbf{y})=0} \pi_i(\mathbf{y}) + \sum_{i:\varphi_i(\mathbf{y})=1} (1-\pi_i(\mathbf{y})) \\
&= \sum_{i:\varphi_i(\mathbf{y})=0}\ \sum_{\mathbf{u}:u_i=1} W(\mathbf{x}(\mathbf{u})|\mathbf{y}) + \sum_{i:\varphi_i(\mathbf{y})=1}\ \sum_{\mathbf{u}:u_i=0} W(\mathbf{x}(\mathbf{u})|\mathbf{y}) \\
&= \sum_{\mathbf{u}}\ \sum_{i:u_i\neq\varphi_i(\mathbf{y})} W(\mathbf{x}(\mathbf{u})|\mathbf{y}) \\
&= \sum_{\mathbf{u}} W(\mathbf{x}(\mathbf{u})|\mathbf{y})d_{\mathrm{H}}(\mathbf{u},\varphi(\mathbf{y})) \\
&= \overline{d}_{\mathrm{H}}(\varphi(\mathbf{y})).
\end{aligned}$$

Q.E.D.

Given $\mathbf{y}$, the vector $(\pi_1,\ldots,\pi_k) = (\pi_1(\mathbf{y}),\ldots,\pi_k(\mathbf{y}))$ can be constructed by the following recursive algorithm.

1. $\sigma := 0$ and $(\pi_1,\ldots,\pi_k) := (0,\ldots,0)$.
2. For all $\mathbf{u} \in \{0,1\}^k$,
   (a) $\omega := Q(\mathbf{u})W(\mathbf{y}|\mathbf{x}(\mathbf{u}))$ and increase σ by ω;
   (b) increase π_i by ω for all $i \in \{1,\ldots,k\}$ such that $u_i = 1$.
3. $\pi_i := \pi_i/\sigma$ for all $i = 1,\ldots,k$.

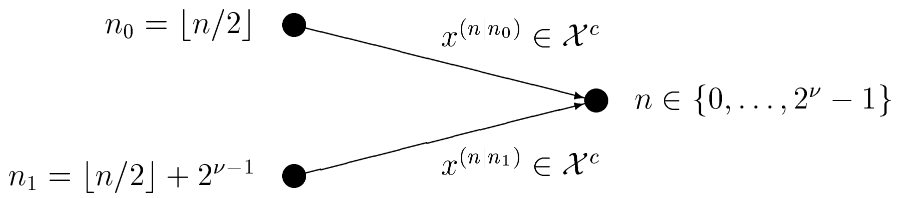

Figure 1: Fragment of the trellis of convolutional code of memory ν for a code of rate $1/c$.

3 Recursive decoding procedure for convolutional codes : when the code trellis is truncated

Let a convolutional code of rate $1/c$ whose encoder has memory ν be used for information transmission. We set that, for all $t = 1, 2, \ldots$, bits $u_{t-\nu+1}, \ldots, u_t$ of information sequence lead encoder to the state $\sum_{\tau=0}^{\nu-1} u_{t-\tau} 2^\tau$, where

$$u_\tau \triangleq 0, \quad \text{for all} \quad \tau \leq 0.$$

Nodes of the code trellis are then identified by integers $0, \ldots, 2^\nu - 1$ in such a way that edges, merging in the node $n \in \{0, \ldots, 2^\nu - 1\}$, go from nodes n_0 and n_1, where

$$n_0 \triangleq \lfloor n/2 \rfloor, \quad n_1 \triangleq \lfloor n/2 \rfloor + 2^{\nu-1}. \tag{3}$$

Let $x^{(n|n_0)}, x^{(n|n_1)} \in \mathcal{X}^c$ denote code subblocks assigned to the edges connecting nodes n_0 and n_1 to node n (see Figure 1) and let any code sequence be constructed as a concatenation of code subblocks assigned to edges of the path in the code trellis corresponding to this sequence. If $t \in \{1, \ldots, \nu\}$, then the only edge entering the node n at level t starts in the node n_0 located at level $t-1$. Note also that, due to the regularity of the code trellis, all paths of length $t \geq \nu + 1$ that enter n and contain node n_b located at level $t - \nu$ have the $(t-\nu)$–th edge corresponding to bit b, i.e.,

$$\mathbf{u}_t \in \mathcal{U}_t^{(n)}, \ u_{t-\nu} = b \implies \mathbf{u}_{t-1} \in \mathcal{U}_{t-1}^{(n_b)} \tag{4}$$

where

$$\mathcal{U}_t^{(n)} \triangleq \Big\{ \mathbf{u}_t = (u_1, \ldots, u_t) : \sum_{\tau=0}^{\nu-1} u_{t-\tau} 2^\tau = n \Big\}. \tag{5}$$

This fact will be used in the decoding algorithm. The goal of the decoder under our considerations is minimization of the bit error probability defined in (1).

Suppose that ν zeroes enter the encoder after the latter one gets T information bits and that the decoder constructs an estimate of the transmitted sequence at the $(T+\nu)$–th level of the code trellis (see Figure 2). Each of 2^T paths in the truncated code trellis can be determined by corresponding information sequence $\mathbf{u}_{T+\nu} \in \{0,1\}^{T+\nu}$ with $u_{T+1} = \ldots = u_{T+\nu} = 0$. In this section we assume that the probability distribution on information sequences is given as

$$Q(\mathbf{u}_{T+\nu}) = \prod_{t=1}^{T} q_t(u_t)$$

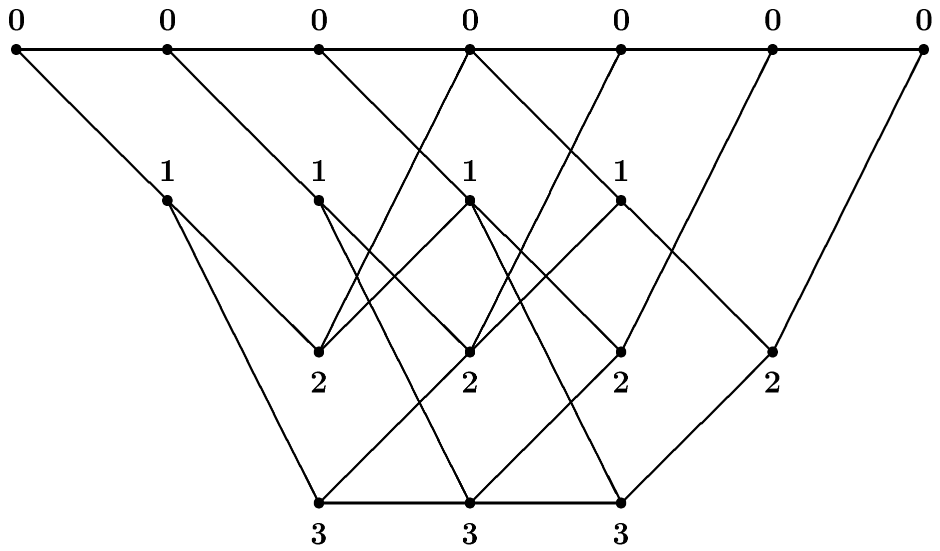

Figure 2: The code trellis of any convolutional encoder of memory $\nu = 2$ and rate $1/c$, which is truncated at level 6.

where $(q_1(0), q_1(1)), \ldots, (q_T(0), q_T(1))$ are known probability distributions on $\{0, 1\}$.

For all $t \in \{1, \ldots, T+\nu\}$, let $y_t \in \mathcal{Y}^c$ denote the t–th received subblock. In foregoing considerations we assume that the codeword $\mathbf{x}(\mathbf{u}_{T+\nu}) \in \mathcal{X}^{T+\nu}$ is transmitted over a discrete memoryless channel in a sense that

$$W(\mathbf{y}_{T+\nu}|\mathbf{x}(\mathbf{u}_{T+\nu})) = \prod_{t=1}^{T+\nu} W(y_t|x(u_{t-\nu}, \ldots, u_t))$$

is the conditional probability to receive the vector $\mathbf{y}_{T+\nu} \in \mathcal{Y}^{T+\nu}$, where $W(y|x), y \in \mathcal{Y}^c$, are given conditional probabilities for all $x \in \mathcal{X}^c$ and $x(u_{t-\nu}, \ldots, u_t) \in \mathcal{X}^c$ denotes the code subblock assigned to the vector $(u_{t-\nu}, \ldots, u_t) \in \{0, 1\}^{\nu+1}$, i.e.,

$$\left.\begin{array}{l} \sum_{\tau=1}^{\nu} u_{t-\tau} 2^\tau = n' \\ \sum_{\tau=0}^{\nu-1} u_{t-\tau} 2^\tau = n \end{array}\right\} \Rightarrow x(u_{t-\nu}, \ldots, u_t) = x^{(n|n')}.$$

We will present a multi–step decoding algorithm designed in such a way that at the t–th step the decoder computes the probability $\omega^{(n)}$ and, if $t \geq \nu + 1$, constructs the rational–valued vector $(\pi_1^{(n)}, \ldots, \pi_{t-\nu}^{(n)})$ for all $n = 0, \ldots, 2^\nu - 1$. The algorithm is given below, where we set

$$q_t^{(n|n_0)} = q_t^{(n|n_1)} = \begin{cases} q_t(0), & \text{if } n \text{ is even and } t \in \{1, \ldots, T\} \\ q_t(1), & \text{if } n \text{ is odd and } t \in \{1, \ldots, T\} \\ 1, & \text{if } 2^{t-T}|n \text{ and } t \in \{T+1, \ldots, T+\nu\} \\ 0, & \text{if } 2^{t-T} \not| n \text{ and } t \in \{T+1, \ldots, T+\nu\} \end{cases}$$

for all $n = 0, \ldots, 2^\nu - 1$.

$1'$. $t := 1$, $\tilde{\omega}^{(0)} := 1$, and $\tilde{\omega}^{(1)} := \ldots := \tilde{\omega}^{(2^\nu - 1)} := 0$.

$2'$. $\sigma := 0$.

$3'$. For all $n = 0, \ldots, 2^\nu - 1$,

(3′ a)

$$\begin{aligned} r^{(n|n_0)} &:= q_t^{(n|n_0)} W(y_t|x^{(n|n_0)}) \\ r^{(n|n_1)} &:= q_t^{(n|n_1)} W(y_t|x^{(n|n_1)}) \\ \omega^{(n)} &:= \tilde{\omega}^{(n_0)} r^{(n|n_0)} + \tilde{\omega}^{(n_1)} r^{(n|n_1)} \\ \sigma &:= \sigma + \omega^{(n)} \end{aligned}$$

(3′ b) if $t \geq \nu + 1$ and $\omega^{(n)} > 0$, then

$$\pi_{t-\nu}^{(n)} := \frac{\tilde{\omega}^{(n_1)} r^{(n|n_1)}}{\omega^{(n)}}$$

(3′ c) if $t > \nu + 1$ and $\omega^{(n)} > 0$, then

$$\pi_\tau^{(n)} := \frac{\tilde{\omega}^{(n_0)} r^{(n|n_0)}}{\omega^{(n)}} \tilde{\pi}_\tau^{(n_0)} + \frac{\tilde{\omega}^{(n_1)} r^{(n|n_1)}}{\omega^{(n)}} \tilde{\pi}_\tau^{(n_1)}$$

for all $\tau = 1, \ldots, t - \nu - 1$.

4′. $\tilde{\omega}^{(n)} := \omega^{(n)}/\sigma$ for all $n = 0, \ldots, 2^\nu - 1$.
If $t \geq \nu + 1$, then $(\tilde{\pi}_1^{(n)}, \ldots, \tilde{\pi}_{t-\nu}^{(n)}) := (\pi_1^{(n)}, \ldots, \pi_{t-\nu}^{(n)})$ for all $n = 0, \ldots, 2^\nu - 1$.

5′. Increase t by 1. If $t \leq T + \nu$, then go to 2′.

6′. End : output $(\pi_1^{(0)}, \ldots, \pi_T^{(0)})$.

Theorem. *For all $n = 0, \ldots, 2^\nu - 1$ and $t = 1, \ldots, T + \nu$, denote the probability $\tilde{\omega}^{(n)}$ computed at step 4′ of the 1′–6′ algorithm by $\tilde{\omega}_t^{(n)}$ and, if $t \geq \nu + 1$, denote the vector $(\pi_1^{(n)}, \ldots, \pi_{t-\nu}^{(n)})$ by $(\pi_{t,1}^{(n)}, \ldots, \pi_{t,t-\nu}^{(n)})$. Then*

$$\tilde{\omega}_t^{(n)} = \frac{Q_t^{(n)}}{S_t} \tag{6}$$

$$\pi_{t,\tau}^{(n)} = \frac{1}{Q_t^{(n)}} \sum_{\mathbf{u}_t \in \mathcal{U}_t^{(n)} : u_\tau = 1} Q(\mathbf{u}_t) W(\mathbf{y}_t | \mathbf{x}(\mathbf{u}_t)), \quad \text{for all } \tau \in \{1, \ldots, t - \nu\} \tag{7}$$

where

$$Q_t^{(n)} \triangleq \sum_{\mathbf{u}_t \in \mathcal{U}_t^{(n)}} Q(\mathbf{u}_t) W(\mathbf{y}_t | \mathbf{x}(\mathbf{u}_t)) \tag{8}$$

$$S_t \triangleq \sum_{N=0}^{2^\nu - 1} Q_t^{(N)}. \tag{9}$$

Proof : It is easy to see that (6) holds for $t = 1$. Let us fix $t \geq 2$ and suppose that (6) holds for all nodes located at level $t - 1$, i.e.,

$$\tilde{\omega}_{t-1}^{(n')} = \frac{Q_{t-1}^{(n')}}{S_{t-1}}, \quad \text{for all } n' = 0, \ldots, 2^\nu - 1 \tag{10}$$

where $Q_{t-1}^{(n')}$ and S_{t-1} are defined by (5), (8), and (9) with $t - 1$ and n' substituted for t and n.

Notice that if $\mathbf{u}_t \in \mathcal{U}_t^{(n)}$, then

$$Q(\mathbf{u}_t)W(\mathbf{y}_t|\mathbf{x}(\mathbf{u}_t)) = Q(\mathbf{u}_{t-1})W(\mathbf{y}_{t-1}|\mathbf{x}(\mathbf{u}_{t-1}))q_t^{(n|n')}W(y_t|x^{(n|n')}) \tag{11}$$

where $n' = \sum_{\tau=0}^{\nu-1} u_{t-1-\tau}2^\tau \in \{n_0, n_1\}$. We take the sum on all $\mathbf{u}_t \in \mathcal{U}_t^{(n)}$ and using notations introduced in step (3′a) obtain

$$Q_t^{(n)} = Q_{t-1}^{(n_0)}r^{(n|n_0)} + Q_{t-1}^{(n_1)}r^{(n|n_1)}. \tag{12}$$

However, by (10) for $n' = n_0, n_1$,

$$Q_{t-1}^{(n_0)} = S_{t-1}\tilde{\omega}_{t-1}^{(n_0)}, \quad Q_{t-1}^{(n_1)} = S_{t-1}\tilde{\omega}_{t-1}^{(n_1)} \tag{13}$$

and (12) implies

$$Q_t^{(n)} = S_{t-1}\Big[\,\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}\,\Big]. \tag{14}$$

Therefore

$$\sum_{N=0}^{2^\nu-1} Q_t^{(N)} = S_{t-1}\sum_{N=0}^{2^\nu-1}\Big[\,\tilde{\omega}_{t-1}^{(N_0)}r^{(N|N_0)} + \tilde{\omega}_{t-1}^{(N_1)}r^{(N|N_1)}\,\Big] \tag{15}$$

where $N_0 \triangleq \lfloor N/2 \rfloor$ and $N_1 \triangleq N_0 + 2^{\nu-1}$. Thus, (6) follows from (14), (15) since $\tilde{\omega}_t^{(n)}$ is computed in the algorithm as

$$\frac{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}{\sum_{N=0}^{2^\nu-1}\Big[\,\tilde{\omega}_{t-1}^{(N_0)}r^{(N|N_0)} + \tilde{\omega}_{t-1}^{(N_1)}r^{(N|N_1)}\,\Big]}$$

To prove (7), let us fix $t \geq \nu + 1$ and using (4), (11) obtain

$$\begin{aligned}
&\sum_{\mathbf{u}_t \in \mathcal{U}_t^{(n)}: u_{t-\nu}=1} Q(\mathbf{u}_t)W(\mathbf{y}_t|\mathbf{x}(\mathbf{u}_t)) \\
&= q_t^{(n|n_1)}W(y_t|x^{(n|n_1)}) \sum_{\mathbf{u}_{t-1} \in \mathcal{U}_{t-1}^{(n_1)}} Q(\mathbf{u}_{t-1})W(\mathbf{y}_{t-1}|\mathbf{x}(\mathbf{u}_{t-1})) \\
&= r^{(n|n_1)}Q_{t-1}^{(n_1)}
\end{aligned}$$

Thus, by (13) and (14),

$$\begin{aligned}
\frac{1}{Q_t^{(n)}} \sum_{\mathbf{u}_t \in \mathcal{U}_t^{(n)}: u_{t-\nu}=1} Q(\mathbf{u}_t)W(\mathbf{y}_t|\mathbf{x}(\mathbf{u}_t)) &= \frac{r^{(n|n_1)}Q_{t-1}^{(n_1)}}{Q_t^{(n)}} \\
&= \frac{\tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}
\end{aligned}$$

The expression on the right–hand side of this equation coincides with the expression in step (3′b) when $\pi_{t-1,t-\nu}^{(n_1)}$ is substituted for $\tilde{\pi}_{t-\nu}^{(n_1)}$.

Let us fix $\tau \in \{1, \ldots, t-\nu-1\}$ and suppose that

$$\pi_{t-1,\tau}^{(n')} = \frac{1}{Q_{t-1}^{(n')}} \sum_{\mathbf{u}_{t-1} \in \mathcal{U}_{t-1}^{(n')}: u_\tau=1} Q(\mathbf{u}_{t-1})W(\mathbf{y}_{t-1}|\mathbf{x}(\mathbf{u}_{t-1})), \quad \text{for all } \ n' = 0, \ldots .2^\nu - 1.$$

Then using (10) and (11) we obtain

$$
\begin{aligned}
&\sum_{\mathbf{u}_t\in\mathcal{U}_t^{(n)}:u_\tau=1} Q(\mathbf{u}_t)W(\mathbf{y}_t|\mathbf{x}(\mathbf{u}_t))\\
&= q_t^{(n|n_0)}W(y_t|x^{(n|n_0)})\sum_{\mathbf{u}_{t-1}\in\mathcal{U}_{t-1}^{(n_0)}:u_\tau=1} Q(\mathbf{u}_{t-1})W(\mathbf{y}_{t-1}|\mathbf{x}(\mathbf{u}_{t-1}))\\
&\quad + q_t^{(n|n_1)}W(y_t|x^{(n|n_1)})\sum_{\mathbf{u}\in\mathcal{U}_{t-1}^{(n_1)}:u_\tau=1} Q(\mathbf{u})W(\mathbf{y}|\mathbf{x}(\mathbf{u}))\\
&= r^{(n|n_0)}Q_{t-1}^{(n_0)}\pi_{t-1,\tau}^{(n_0)} + r^{(n|n_1)}Q_{t-1}^{(n_1)}\pi_{t-1,\tau}^{(n_1)}\\
&= S_{t-1}\Big[\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)}\pi_{t-1,\tau}^{(n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}\pi_{t-1,\tau}^{(n_1)}\Big].
\end{aligned}
\tag{16}
$$

By (14),

$$
S_{t-1} = \frac{Q_t^{(n)}}{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}
$$

and (16) implies

$$
\begin{aligned}
&\frac{1}{Q_t^{(n)}}\sum_{\mathbf{u}_t\in\mathcal{U}_t^{(n)}:u_\tau=1} Q(\mathbf{u}_t)W(\mathbf{y}_t|\mathbf{x}(\mathbf{u}_t))\\
&= \frac{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)}}{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}\pi_{t-1,\tau}^{(n_0)} + \frac{\tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}{\tilde{\omega}_{t-1}^{(n_0)}r^{(n|n_0)} + \tilde{\omega}_{t-1}^{(n_1)}r^{(n|n_1)}}\pi_{t-1,\tau}^{(n_1)}
\end{aligned}
$$

The sum on the right–hand side of this equation coincides with the sum in step ($3'$c) of the $1'$–$6'$ algorithm when $\pi_{t-1,\tau}^{(n_0)}$ and $\pi_{t-1,\tau}^{(n_1)}$ are substituted for $\tilde{\pi}_\tau^{(n_0)}$ and $\tilde{\pi}_\tau^{(n_1)}$.

Thus, (6) and (7) have been proven by induction.

Q.E.D.

It is interesting to notice that the MAP decoding, which is known as the Viterbi decoding if the probability distribution on information sequences is uniform, i.e., $q_t(0) = q_t(1) = 1/2$ for all $t = 1,\dots,T$, can be represented as a result of modification of the $1'$–$6'$ algorithm. Namely, $(\pi_1^{(n)},\dots,\pi_{t-\nu}^{(n)})$ are binary vectors in this case and steps ($3'$b), ($3'$c) are replaced with the steps (3b), (3c) given below.

(3b) if $t \ge \nu+1$ and $\tilde{\omega}^{(n)} > 0$, then

$$\pi_{t-\nu}^{(n)} := b$$

where

$$
b = \begin{cases} 0, & \text{if } \tilde{\omega}^{(n_0)}r^{(n|n_0)} \ge \tilde{\omega}^{(n_1)}r^{(n|n_1)}\\ 1, & \text{if } \tilde{\omega}^{(n_0)}r^{(n|n_0)} < \tilde{\omega}^{(n_1)}r^{(n|n_1)} \end{cases}
$$

(3c) if $t > \nu+1$ and $\tilde{\omega}^{(n)} > 0$, then

$$(\pi_1^{(n)},\dots,\pi_{t-\nu-1}^{(n)}) := (\tilde{\pi}_1^{(n_b)},\dots,\tilde{\pi}_{t-\nu-1}^{(n_b)}).$$

For the notations to be more conventional, we will call this algorithm "the Viterbi decoding", and the algorithm consisting of steps $1'$–$6'$ "the Viterbi$'$ decoding". Data in Table 1 illustrate these algorithms for a specific example, where the $(5,7)$–encoder has memory $\nu = 2$, rate 1/2 and

$$x(u_{t-2},u_{t-1},u_t) = (u_{t-2}\oplus u_{t-1}, u_{t-2}\oplus u_{t-1}\oplus u_t).$$

Table 1: The vectors $(\pi_1^{(n)}, \ldots, \pi_{t-\nu}^{(n)})$ formed by the Viterbi and Viterbi′ decoding algorithms for the (5,7)–encoder when the code trellis is truncated at level 6; the code sequence is transmitted over a BSC with crossover probability 0.25 and $\mathbf{y}$ = (01, 11, 01, 00, 10, 01) is the received sequence.

		Viterbi				Viterbi′			
t	n	$\pi_1^{(n)}$	$\pi_2^{(n)}$	$\pi_3^{(n)}$	$\pi_4^{(n)}$	$\pi_1^{(n)}$	$\pi_2^{(n)}$	$\pi_3^{(n)}$	$\pi_4^{(n)}$
3	**0**	1				0.750			
	1	1				0.750			
	2	0				0.036			
	3	1				0.750			
4	**0**	1	0			0.438	0.438		
	1	0	1			0.047	0.984		
	2	1	1			0.750	0.750		
	3	1	1			0.750	0.750		
5	**0**	1	0	0		0.571	0.571	0.429	
	2	0	1	0		0.488	0.837	0.628	
6	**0**	0	1	0	1	0.516	0.750	0.562	0.672

4 Recursive source–channel decoding procedure for convolutional codes when the code trellis is truncated

Suppose that there is a discrete source whose messages are chosen from a finite set $\mathcal{M}$. The sequence of messages is mapped to the code sequence constructed as a concatenation of codewords of a prefix binary tree code. For example, let $\mathcal{M} = \{1, 2, 3\}$ and let the source code be defined as

$$(\mathbf{u}(1), \mathbf{u}(2), \mathbf{u}(3)) = (00, 11, 1001). \tag{17}$$

The code tree is given in Figure 3.

Let us assign code bits to edges of the tree and associate capital letters with the nodes in such a way that **S** is associated both with the root and with all terminal nodes. All other letters identify corresponding nodes. We consider these letters as states of the source encoder and interpret encoding of any sequence of messages as passing of the encoder through the nodes of the code trellis given in Figure 4. Let

$$\mathcal{S} = \{\,\mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\,\}$$

denote the set of possible states. The trellis of any source encoder, which uses a binary prefix tree code, can be constructed in a similar way [3].

Consider the following data transmission situation. Suppose that the source is allowed to generate only sequences of messages, which are mapped to code sequences of length T. The code sequence enters a convolutional encoder of memory ν whose trellis is truncated at level $T + \nu$. Then the structure of the source encoder puts constraints on the set of possible information sequences for the convolutional encoder and determines

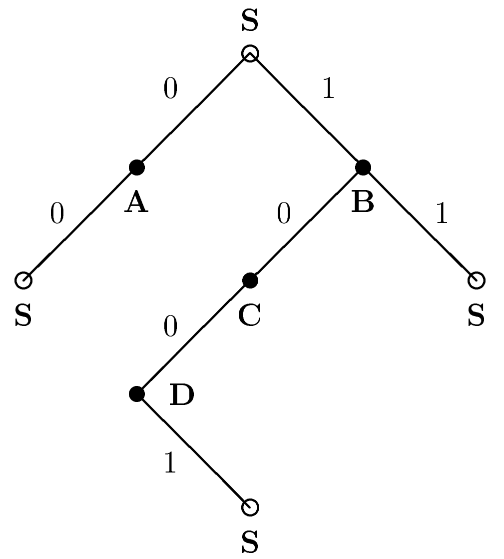

Figure 3: Tree of the source code (17).

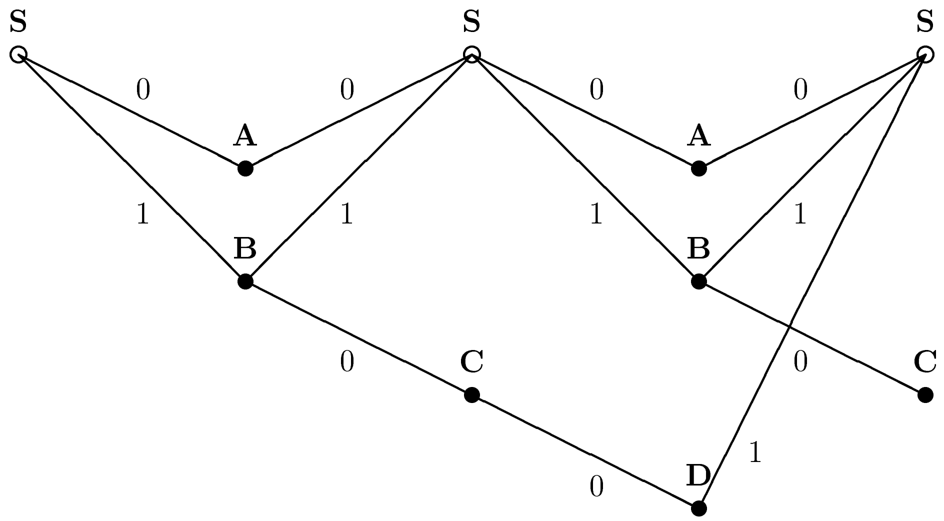

Figure 4: First 4 levels of the trellis of the encoder for the code (17).

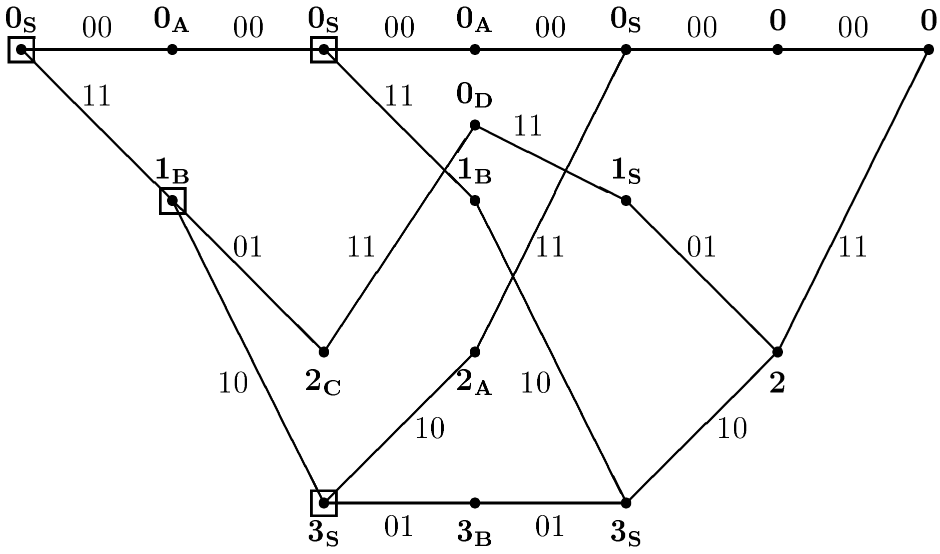

Figure 5: The joint source–channel trellis generated by the truncated trellis of the (5,7)–convolutional encoder and the trellis of the source encoder given in Figure 3.

the probability distribution on these sequences. The decoding algorithm in this case can be described using the joint trellis constructed as a Cartesian product of the trellis of the source encoder and the trellis of convolutional encoder (see Figure 5 and Table 2). We identify nodes of the joint trellis by n_β, where $n \in \{0, \ldots, 2^\nu - 1\}$ and $\beta \in \mathcal{S}$ in such a way that the node n_β is located at level t if and only if the node β is located at level t of the trellis of the source encoder and $\mathbf{u} \in \mathcal{U}_t^{(n)}$, where set $\mathcal{U}_t^{(n)}$ is defined in (5), is the corresponding code sequence.

Suppose that the sequence of $T+\nu$ code subblocks, associated with edges of the joint trellis, is transmitted over a channel. The decoder has to find a sequence of messages, which is mapped to the code sequence of length T having the smallest bit error probability relative to the received sequence $\mathbf{y}$ of length $(T+\nu)c$. This problem can be solved using the $1'$–$6'$ algorithm if we specify a probability distribution on sequences that can be constructed at the output of the source encoder. One of possible assignments is as follows. Let

$$P(m_1, \ldots, m_K) \triangleq \prod_{k=1}^{K} p(m_k)$$

for all $\mathbf{m} = (m_1, \ldots, m_K) \in \mathcal{M}^K$ and all $K \geq 1$, where $(p(m), m \in \mathcal{M})$ is a given probability distribution. Furthermore, let $\ell(m)$ denote the length of the codeword for the message $m \in \mathcal{M}$. Then

$$\mathcal{M}_T \triangleq \bigcup_{K \geq 1} \Big\{ (m_1, \ldots, m_K) \in \mathcal{M}^K : \ell(m_1) + \ldots + \ell(m_K) = T \Big\}$$

is the set of possible sequences of messages for the system of data transmission under considerations. For all $t \in \{1, \ldots, T\}$ and nodes $n_\beta, n'_{\beta'}$ of the joint trellis located at levels t and $t-1$, respectively, denote

$$q_t^{(n_\beta | n'_{\beta'})} \triangleq \frac{\sum_{\mathbf{m} \in \mathcal{M}_{T,t}^{(n_\beta)}} P(\mathbf{m})}{\sum_{\mathbf{m} \in \mathcal{M}_{T,t-1}^{(n'_{\beta'})}} P(\mathbf{m})}$$

Table 2: Paths in the trellis in Figure 5 and corresponding codewords.

$\mathbf{m}$	$\beta_0, \ldots, \beta_4$	$\mathbf{u}_4$	$\mathbf{x}(\mathbf{u}_4)$					
11	**SASAS**	0000	00	00	00	00	00	00
12	**SASBS**	0011	00	00	11	10	10	11
3	**SBCDS**	1001	11	01	11	11	01	11
21	**SBSAS**	1100	11	10	10	11	00	00
22	**SBSBS**	1111	11	10	01	01	10	11

where $\mathcal{M}_{T,t}^{(n_\beta)}$ is the set of messages such that the state of the source encoder at level t is equal to β and the sequence $\mathbf{u}_t$ belongs to the set $\mathcal{U}_t^{(n)}$; the set $\mathcal{M}_{T,t-1}^{(n'_{\beta'})}$ is defined similarly.

Let us illustrate notations above for the code (17) and $T = 4$. Let

$$(p(1), p(2), p(3)) = (\sqrt{2}-1,\ \sqrt{2}-1,\ 3-2\sqrt{2}). \tag{18}$$

Then all 5 paths in the infinite code trellis of the source encoder, which go from the state $\mathbf{S}$ at level t to the state $\mathbf{S}$ at level $t+4$ are equally likely since $(\sqrt{2}-1)^2 = 3-2\sqrt{2}$. However, truncation of the trellis of the convolutional encoder at level $T+\nu$ implies

$$\begin{aligned}
q_1^{(\mathbf{1}_{\mathrm{B}}|\mathbf{0}_{\mathrm{S}})} &= \frac{P(3)+P(21)+P(22)}{P(11)+P(12)+P(3)+P(21)+P(22)} = 0.6\\
q_2^{(\mathbf{3}_{\mathrm{S}}|\mathbf{1}_{\mathrm{B}})} &= \frac{P(21)+P(22)}{P(3)+P(21)+P(22)} = 0.667\\
q_3^{(\mathbf{1}_{\mathrm{B}}|\mathbf{0}_{\mathrm{S}})} &= \frac{P(21)+P(22)}{P(11)+P(12)+P(21)+P(22)} = 0.5\\
q_3^{(\mathbf{3}_{\mathrm{B}}|\mathbf{3}_{\mathrm{S}})} &= \frac{P(12)+P(22)}{P(11)+P(21)+P(12)+P(22)} = 0.5.
\end{aligned}$$

Current results of the $1'$–$6'$ algorithm for this example are shown in Table 3, and calculations bringing these results for $t = 4$, $y_4 = 00$ are presented below.

$$\begin{array}{rl}
n = \mathbf{0}_{\mathrm{S}} & \\
n_0 = \mathbf{0}_{\mathrm{A}}: & \tilde{\omega}^{(n_0)} r^{(n|n_0)} = 0.297 \cdot 1 \cdot (0.75 \cdot 0.75) = 0.1671\\
n_1 = \mathbf{2}_{\mathrm{A}}: & \tilde{\omega}^{(n_1)} r^{(n|n_1)} = 0.031 \cdot 1 \cdot (0.25 \cdot 0.25) = 0.0019\\
& \pi_1^{(n)} = (0.1671/0.1690) \cdot 0 + (0.0019/0.1690) \cdot 1 = 0.011\\
& \pi_2^{(n)} = 0.0019/0.1690 = 0.011\\
\hline
n = \mathbf{1}_{\mathrm{S}} & \\
n_0 = \mathbf{0}_{\mathrm{D}}: & \tilde{\omega}^{(n_0)} r^{(n|n_0)} = 0.094 \cdot 1 \cdot (0.25 \cdot 0.25) = 0.0059\\
& \tilde{\omega}^{(n_1)} r^{(n|n_1)} = 0\\
& \pi_1^{(n)} = (0.0059/0.0059) \cdot 1 = 1\\
& \pi_2^{(n)} = 0/0.0059 = 0\\
\hline
n = \mathbf{3}_{\mathrm{S}} & \\
n_0 = \mathbf{1}_{\mathrm{B}}: & \tilde{\omega}^{(n_0)} r^{(n|n_0)} = 0.297 \cdot 1 \cdot (0.25 \cdot 0.75) = 0.0557\\
n_1 = \mathbf{3}_{\mathrm{B}}: & \tilde{\omega}^{(n_1)} r^{(n|n_1)} = 0.281 \cdot 1 \cdot (0.75 \cdot 0.25) = 0.0527\\
& \pi_1^{(n)} = (0.0557/0.1084) \cdot 0 + (0.0527/0.1084) \cdot 1 = 0.486\\
& \pi_2^{(n)} = 0.0527/0.1084 = 0.486
\end{array}$$

Table 3: Data formed by the Viterbi′ decoder for the trellis in Figure 5; $\mathbf{y}$ = (01, 11, 01, 00, 10, 01) is the received sequence and 0.25 is the crossover probability of a BSC.

t	n_β	$\tilde{\omega}_t^{(n_\beta)}$	$\pi_1^{(n_\beta)}$	$\pi_2^{(n_\beta)}$	$\pi_3^{(n_\beta)}$	$\pi_4^{(n_\beta)}$
1	$\mathbf{0}_{\mathrm{A}}$	0.414				
	$\mathbf{1}_{\mathrm{B}}$	0.586				
2	$\mathbf{0}_{\mathrm{S}}$	0.679				
	$\mathbf{2}_{\mathrm{C}}$	0.107				
	$\mathbf{3}_{\mathrm{S}}$	0.214				
3	$\mathbf{0}_{\mathrm{A}}$	0.297	0			
	$\mathbf{0}_{\mathrm{D}}$	0.094	1			
	$\mathbf{1}_{\mathrm{B}}$	0.297	0			
	$\mathbf{2}_{\mathrm{A}}$	0.031	1			
	$\mathbf{3}_{\mathrm{B}}$	0.281	1			
4	$\mathbf{0}_{\mathrm{S}}$	0.596	0.011	0.011		
	$\mathbf{1}_{\mathrm{S}}$	0.021	1	0		
	$\mathbf{3}_{\mathrm{S}}$	0.383	0.486	0.486		
5	$\mathbf{0}$	0.340	0.011	0.011	0	
	$\mathbf{2}$	0.660	0.495	0.483	0.994	
6	$\mathbf{0}$	1	0.330	0.322	0.656	0.660

Since $\sigma = 0.1690 + 0.0059 + 0.1084 = 0.2833$, we also have

$$\tilde{\omega}_4^{(n)} = \begin{cases} 0.1690/0.2833 = 0.596, & \text{if } n = \mathbf{0}_{\mathrm{S}} \\ 0.0059/0.2833 = 0.021, & \text{if } n = \mathbf{1}_{\mathrm{S}} \\ 0.1084/0.2833 = 0.383, & \text{if } n = \mathbf{3}_{\mathrm{S}}. \end{cases}$$

If the vector

$$(\pi_1(\mathbf{y}), \ldots, \pi_T(\mathbf{y})) = (0.330, 0.322, 0.656, 0.660)$$

constructed at the output of the decoder is transformed to the binary vector using (2), then we obtain the vector

$$(\hat{u}_1, \ldots, \hat{u}_T) = (0, 0, 1, 1).$$

This vector is a concatenation of the codewords of the source encoder for the sequence of messages $(1, 2)$. Thus, by Proposition, $(1, 2)$ is the estimate of the sequence of messages minimizing the bit error probability. In general case, the vector $(\pi_1(\mathbf{y}), \ldots, \pi_T(\mathbf{y}))$ should be treated as the input vector to the Viterbi decoder, which is searching for the closest path in the truncated code trellis of the source encoder consisting of T levels. Each path starts in the node $\mathbf{S}$ at level 0 and ends in the node $\mathbf{S}$ at level T (see Figure 4 where the edge connecting nodes $\mathbf{B}$ and $\mathbf{C}$ at level 4 has to be removed). If $x \in \{0, 1\}$ is the code bit associated with some edge of the trellis going to level t, then the metric of this edge is defined as $x - \pi_t(\mathbf{y})$ and the metric of any path in the trellis is defined as the sum of metrics of its edges.

5 Recursive decoding procedure for convolutional codes when the code trellis is not truncated

Truncation of the code trellis at level $T+\nu$ allows us to treat convolutional code as a block code of length $(T+\nu)c$ and rate $T/(Tc+\nu c)$. Contrary, "a convolutional approach" is based on continuous decoding when the decoder makes a decision about the first bit of the transmitted information sequence at level t^*, decision about the second bit of the transmitted information sequence at level t^*+1, etc. Similar requirement can be presented for the decoder minimizing the bit error probability. The $1'$–$6'$ algorithm can be extended in this direction when the decoder forms the best estimate of the t–th transmitted bit based on the analysis of information sequences belonging to the set

$$\mathcal{U}'_t \triangleq \bigcup_{n=0}^{2^\nu-1} \mathcal{U}_{t+t^*}^{(n)}$$

where $\mathcal{U}_{t+t^*}^{(0)},\ldots,\mathcal{U}_{t+t^*}^{(2^\nu-1)}$ are defined by (5) with the index $t+t^*$ substituted for t. Note that the decoding algorithm, given below, can be also extended to the situation of joint source–channel decoding considered in the previous section.

1^*. $t:=1$, $\tilde{\omega}^{(0)}:=1$, and $\tilde{\omega}^{(1)}:=\ldots:=\tilde{\omega}^{(2^\nu-1)}:=0$.

2^*. $\sigma:=0$.

3^*. For all $n=0,\ldots,2^\nu-1$,

(3^* a)

$$\begin{aligned} r^{(n|n_0)} &:= q_t^{(n|n_0)} W(y_t|x^{(n|n_0)}) \\ r^{(n|n_1)} &:= q_t^{(n|n_1)} W(y_t|x^{(n|n_1)}) \\ \omega^{(n)} &:= \tilde{\omega}^{(n_0)} r^{(n|n_0)} + \tilde{\omega}^{(n_1)} r^{(n|n_1)} \\ \sigma &:= \sigma+\omega^{(n)} \end{aligned}$$

(3^* b) if $t\ge\nu+1$ and $\omega^{(n)}>0$, then

$$\pi_{t^*-\nu}^{(n)} := \frac{\tilde{\omega}^{(n_1)} r^{(n|n_1)}}{\omega^{(n)}}$$

(3^* c) if $t>\nu+1$ and $\omega^{(n)}>0$, then

$$\pi_\tau^{(n)} := \frac{\tilde{\omega}^{(n_0)} r^{(n|n_0)}}{\omega^{(n)}}\tilde{\pi}_\tau^{(n_0)} + \frac{\tilde{\omega}^{(n_1)} r^{(n|n_1)}}{\omega^{(n)}}\tilde{\pi}_\tau^{(n_1)}$$

for all $\tau=1,\ldots,t^*-\nu-1$.

4^*. $\tilde{\omega}^{(n)}:=\omega^{(n)}/\sigma$ for all $n=0,\ldots,2^\nu-1$.
Output

$$\hat{\pi}_{t-t^*} := \sum_{n=0}^{2^\nu-1} \tilde{\omega}^{(n)}\pi_1^{(n)}.$$

If $t\ge\nu+1$, then $(\tilde{\pi}_1^{(n)},\ldots,\tilde{\pi}_{t^*-\nu-1}^{(n)}) := (\pi_2^{(n)},\ldots,\pi_{t^*-\nu}^{(n)})$ for all $n=0,\ldots,2^\nu-1$.

5^*. Increase t by 1. If $t\le T^*$, then go to 2^*.

6^*. End.

References

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 3, pp. 284–287, Mar. 1974.

[2] A. Abedi, P. Chaudhari, and A. K. Khandani, "On some properties of the bit decoding algorithms," *Proc. 7–th Canadian Workshop on Information Theory*, Vancouver, British Columbia, pp. 106–109, June 2001.

[3] V. B. Balakirsky, "Joint source–channel coding with variable length codes," *Probl. Inform. Transmission*, vol. 37, no. 1, pp. 12–27, Jan.–Mar. 2001.
English translation : pp. 10–23 (ISSN : 0032–9460).