Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 95360, Pages 1–15
DOI 10.1155/ASP/2006/95360

# A Systematic Approach to Modified BCJR MAP Algorithms for Convolutional Codes

**Sichun Wang**[1] **and François Patenaude**[2]

[1] *Defence Research and Development Canada – Ottawa, Ottawa, ON, Canada K1A 0Z4*
[2] *Communications Research Centre Canada, Ottawa, ON, Canada K2H 8S2*

Received 19 November 2004; Revised 19 July 2005; Accepted 12 September 2005

Recommended for Publication by Vincent Poor

Since Berrou, Glavieux and Thitimajshima published their landmark paper in 1993, different modified BCJR MAP algorithms have appeared in the literature. The existence of a relatively large number of similar but different modified BCJR MAP algorithms, derived using the Markov chain properties of convolutional codes, naturally leads to the following questions. What is the relationship among the different modified BCJR MAP algorithms? What are their relative performance, computational complexities, and memory requirements? In this paper, we answer these questions. We derive systematically four major modified BCJR MAP algorithms from the BCJR MAP algorithm using simple mathematical transformations. The connections between the original and the four modified BCJR MAP algorithms are established. A detailed analysis of the different modified BCJR MAP algorithms shows that they have identical computational complexities and memory requirements. Computer simulations demonstrate that the four modified BCJR MAP algorithms all have identical performance to the BCJR MAP algorithm.

## 1. INTRODUCTION

In 1993, Berrou et al. [1] introduced new types of codes, called turbo codes, which have demonstrated performance close to the theoretical limit predicted by information theory [2]. In the iterative decoding strategy for turbo codes, a soft-input soft-output (SISO) MAP algorithm is used to perform the decoding operation for the two constituent recursive systematic convolutional codes (RSC). The SISO MAP algorithm presented in [1], which is called the BGT MAP algorithm in [3], is a modified version of the BCJR MAP algorithm proposed in [4]. The BGT MAP algorithm formally appears very complicated. Later, Pietrobon and Barbulescu derived a simpler modified BCJR MAP algorithm [5], which is called the PB MAP algorithm [3]. However, the PB MAP algorithm is not a direct simplification of the BGT MAP algorithm, even though they share similar structures. In [3], the BGT MAP algorithm is directly simplified to obtain a new modified BCJR MAP algorithm that keeps the structure of the BGT MAP algorithm but uses simpler recursive procedures. This new modified BCJR MAP algorithm is called the SBGT MAP algorithm in [3]. The main difference between the SBGT and BGT MAP algorithms lies in the fact that for the BGT MAP algorithm, the forward and backward recursions (cf. [1, equations (21) and (22)]) are formulated in such a way that redundant divisions are involved, whereas in the SBGT MAP algorithm, these redundant computations are removed.

In [3], it is also shown that the symmetry of the trellis diagram of an RSC code can be utilized (albeit implicitly) to derive another modified BCJR MAP algorithm which possesses a structure that is dual to that of the SBGT MAP algorithm and has the same signal processing and memory requirements. This new modified BCJR MAP algorithm is called the dual SBGT MAP algorithm in [3]. The Dual SBGT MAP algorithm will be called the DSBGT MAP algorithm in this paper.

The BCJR and the modified BCJR MAP algorithms are all derived from first principles by utilizing the Markov chain properties of convolutional codes. Some of the modified BCJR MAP algorithms, as well as the BCJR itself, have actually been implemented in hardware. From both theoretical and practical perspectives, it is of great interest and importance to acquire an understanding of the exact relationship among the different modified BCJR MAP algorithms and their relative advantages.

In this paper, we first derive the BCJR MAP algorithm from first principles for a rate $1/n$ recursive systematic convolutional code, where $n \geq 2$ is any positive integer. We then systematically derive the aforementioned modified BCJR MAP algorithms and a dual version of the PB MAP algorithm from the BCJR MAP algorithm using simple mathematical transformations. By doing this, we succeed in establishing simple connections among these algorithms. In particular, we show that the modified BCJR MAP algorithm of Pietrobon and Barbulescu can be directly derived from the SBGT MAP algorithm via two simple permutations.

A detailed analysis of the BCJR and the four modified BCJR MAP algorithms formulated in this paper shows that they all have identical computational complexities and memory requirements when implemented appropriately. Systematic computer simulations demonstrate that the four modified BCJR MAP algorithms all have identical performance to the BCJR MAP algorithm.

This paper is organized as follows. In Section 2, the now classical BCJR MAP algorithm is revisited and the notation and terminology used in this paper are introduced. In Section 3, it is shown how the SBGT MAP algorithm can be derived from the BCJR MAP algorithm. In Section 4, a dual version of the SBGT MAP algorithm (the dual SBGT MAP algorithm or the DSBGT MAP algorithm) is derived from the BCJR MAP algorithm. In Section 5, it is shown how the PB MAP algorithm of Pietrobon and Barbulescu can be directly derived from the SBGT MAP algorithm by performing simple permutations on the nodes of the trellis diagram of an RSC code. In Section 6, by performing similar permutations, a new modified BCJR MAP algorithm, called the DPB MAP algorithm in this paper, is derived from the DSBGT MAP algorithm. The DPB MAP algorithm can be considered a dual version of the modified BCJR MAP algorithm of Pietrobon and Barbulescu presented in Section 5. In Section 7, a detailed comparative analysis of computational complexities and memory requirements is carried out, where the BCJR and the four modified BCJR MAP algorithms are shown to have the same computational complexities and memory requirements. In Section 8, computer simulations are discussed, which were performed for the rate 1/2 and rate 1/3 turbo codes defined in the CDMA2000 standard using the BCJR, SBGT, DSBGT, PB, and DPB MAP algorithms. As expected, under identical simulation conditions, the BCJR and the four modified BCJR MAP algorithms formulated here all have identical BER (bit error rate) and FER (frame error rate) performance. Finally, Section 9 concludes this paper.

## 2. THE BCJR MAP ALGORITHM REVISITED

To characterize the precise relationship between the original BCJR MAP algorithm and the modified BCJR MAP algorithms, we will present a detailed derivation of the original BCJR MAP algorithm in this section and, in doing so, set up the notation and terminology of this paper. Our derivations show that a proper initialization of the $\beta$ sequence in the BCJR MAP algorithm in fact does not require any a priori assumptions on the final state of the recursive systematic convolutional code. In other words, no information on the final encoder state is required in the derivation of the original BCJR MAP algorithm. This statement also holds true for the modified BCJR MAP algorithms. Note that in [4], it is assumed that the final encoder state is the all-zero state.

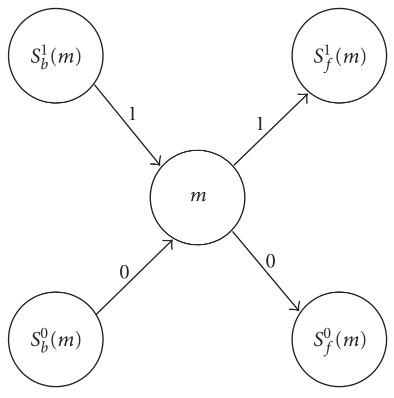

FIGURE 1: Transition diagram of an RSC code.

Let $n \geq 2$, $\nu \geq 1$, $\tau \geq 1$ be positive integers and consider a rate $1/n$ constraint length $\nu + 1$ binary recursive systematic convolutional (RSC) code. Given an input data bit $i$ and an encoder state $m$, the rate $1/n$ RSC encoder makes a state transition from state $m$ to a unique new state $S$ and produces an $n$-bit codeword $\mathbf{X}$. The new encoder state $S$ will be denoted by $S_f^i(m)$, $i = 0, 1$. The $n$ bits of the codeword $\mathbf{X}$ consist of the systematic data bit $i$ and $n - 1$ parity check bits. These $n - 1$ parity check bits will be denoted, respectively, by $Y_1(i, m), Y_2(i, m), \ldots, Y_{n-1}(i, m)$. On the other hand, there is a unique encoder state $T$ from which the encoder makes a state transition to the state $m$ for an input bit $i$. The encoder state $T$ will be denoted by $S_b^i(m)$, $i = 0, 1$. The relationship among the encoder state $m$ and the encoder states $S_b^0(m)$, $S_b^1(m)$, $S_f^0(m)$, and $S_f^1(m)$ is depicted by the state transition diagram in Figure 1. It can be verified that each of the four mappings $S_b^0 : m \to S_b^0(m)$, $S_b^1 : m \to S_b^1(m)$, $S_f^0 : m \to S_f^0(m)$, and $S_f^1 : m \to S_f^1(m)$ is a one-to-one correspondence from the set $\mathcal{M} = \{0, 1, \ldots, 2^\nu - 1\}$ onto itself. In other words, each of the four mappings $S_b^0$, $S_b^1$, $S_f^0$, and $S_f^1$ is a permutation from $\mathcal{M} = \{0, 1, \ldots, 2^\nu - 1\}$ onto itself. It can be verified that $S_f^i(S_b^i(m)) = m$ and $S_b^i(S_f^i(m)) = m$ for $i = 0, 1$, $m = 0, 1, \ldots, 2^\nu - 1$.

Assume the encoder starts at the all-zero state $S_0 = 0$ and encodes a sequence of information data bits $d_1, d_2, d_3, \ldots, d_\tau$. At time $t$, the input into the encoder is $d_t$, which induces the encoder state transition from $S_{t-1}$ to $S_t$ and generates an $n$-bit codeword (vector) $X_t$. The codewords $X_t$ are BPSK modulated and transmitted through an AWGN channel. The matched filter at the receiver yields a sequence of noisy sample vectors $Y_t = 2X_t - \mathbf{1} + \mathcal{N}_t$, $t = 1, 2, 3, \ldots, \tau$, where $\mathbf{1}$ is the $n$-dimensional vector with all its components equal to 1, $X_t$ is an $n$-bit codeword consisting of zeros and ones, and $\mathcal{N}_t$ is an $n$-dimensional random vector with i.i.d. zero-mean Gaussian noise components with variance $\sigma^2 > 0$. Since there are $\nu \geq 1$ memory cells in the RSC encoder, there are $M = 2^\nu$ encoder states, represented by the nonnegative integers $m = 0, 1, 2, \ldots, 2^\nu - 1$. Let

$$\begin{aligned} Y_1^t &= (Y_1, \ldots, Y_t), \quad 1 \le t \le \tau, \\ Y_{t+1}^\tau &= (Y_{t+1}, \ldots, Y_\tau), \quad 1 \le t \le \tau - 1, \\ Y_t &= \left(r_t^{(1)}, r_t^{(2)}, \ldots, r_t^{(n)}\right), \quad 1 \le t \le \tau, \end{aligned} \tag{1}$$

where $r_t^{(1)}$ is the matched filter output sample generated by the systematic data bit $d_t$ and $r_t^{(2)}, \ldots, r_t^{(n)}$ are matched filter output samples generated by the $n - 1$ parity check bits $Y_1(d_t, S_{t-1}), \ldots, Y_{n-1}(d_t, S_{t-1})$, respectively. Let

$$\Lambda(d_t) = \log \frac{\Pr\{d_t = 1 \mid Y_1^\tau\}}{\Pr\{d_t = 0 \mid Y_1^\tau\}}, \quad 1 \le t \le \tau, \tag{2}$$

$$L_a(d_t) = \log \frac{\Pr\{d_t = 1\}}{\Pr\{d_t = 0\}}, \quad 1 \le t \le \tau. \tag{3}$$

$\Lambda(d_t)$ and $L_a(d_t)$ are called, respectively, the a posteriori probability (APP) sequence and the a priori information sequence of the input data sequence $d_t$. In the first half iteration of the turbo decoder, $L_a(d_t) = 0$, since the input data sequence $d_t$ is assumed i.i.d.

The BCJR MAP algorithm centres around the computation of the following joint probabilities:

$$\begin{aligned} \lambda_t(m) &= \Pr\{S_t = m;\ Y_1^\tau\}, \\ \sigma_t(m', m) &= \Pr\{S_{t-1} = m';\ S_t = m;\ Y_1^\tau\}, \end{aligned} \tag{4}$$

where $1 \le t \le \tau$ and $0 \le m', m \le 2^\nu - 1$.

To compute $\lambda_t(m)$ and $\sigma_t(m', m)$, let us define the probability sequences

$$\begin{aligned} \alpha_t(m) &= \Pr\{S_t = m;\ Y_1^t\}, \quad 1 \le t \le \tau, \\ \beta_t(m) &= \Pr\{Y_{t+1}^\tau \mid S_t = m\}, \quad 1 \le t \le \tau - 1, \\ \gamma_t(m', m) &= \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\}, \quad 1 \le t \le \tau, \\ \gamma_i(Y_t, m', m) &= \Pr\{d_t = i;\ S_t = m;\ Y_t \mid S_{t-1} = m'\}, \\ & i = 0, 1,\ 1 \le t \le \tau. \end{aligned} \tag{5}$$

At this stage, it is important to emphasize that $\beta_\tau(m)$ and $\alpha_0(m)$ are not yet defined. In other words, the boundary conditions or initial values for the backward and forward recursions are undetermined. The boundary values (initial conditions) will be determined shortly from the inherent logical consistency among the computed probabilities.

Now assume that $1 \le t \le \tau - 1$. We have

$$\begin{aligned} \lambda_t(m) &= \Pr\{S_t = m;\ Y_1^\tau\} \\ &= \Pr\{S_t = m;\ Y_1^t;\ Y_{t+1}^\tau\} \\ &= \Pr\{S_t = m;\ Y_1^t\} \Pr\{Y_{t+1}^\tau \mid S_t = m;\ Y_1^t\} \\ &= \Pr\{S_t = m;\ Y_1^t\} \Pr\{Y_{t+1}^\tau \mid S_t = m\} \\ &= \alpha_t(m)\beta_t(m). \end{aligned} \tag{6}$$

Here we used the equality

$$\Pr\{Y_{t+1}^\tau \mid S_t = m;\ Y_1^t\} = \Pr\{Y_{t+1}^\tau \mid S_t = m\}, \tag{7}$$

which follows from the Markov chain property that if $S_t$ is known, events after time $t$ do not depend on $Y_1^t$. Similar facts are used in a number of places in this paper. The reader is referred to [6] for more detailed discussions on Markov chains.

Now let $t = \tau$. We have

$$\begin{aligned} \lambda_\tau(m) &= \Pr\{S_t = m;\ Y_1^\tau\} = \Pr\{S_t = m;\ Y_1^t\} \\ &= \alpha_t(m) \times 1 = \alpha_t(m)\beta_t(m). \end{aligned} \tag{8}$$

Here for the first time, we have defined $\beta_\tau(m) = 1$, $m = 0, 1, \ldots, 2^\nu - 1$. Note that $\beta_\tau(m)$ was not defined in (5).

It can be shown that $\sigma_t(m', m)$ can be expressed in terms of the $\alpha$, $\beta$, and $\gamma$ sequences. In fact, if $2 \le t \le \tau - 1$, we have

$$\begin{aligned} \sigma_t(m', m) &= \Pr\{S_{t-1} = m';\ S_t = m;\ Y_1^\tau\} \\ &= \Pr\{S_{t-1} = m';\ Y_1^{t-1};\ S_t = m;\ Y_t;\ Y_{t+1}^\tau\} \\ &= \Pr\{S_{t-1} = m';\ Y_1^{t-1}\} \\ &\quad \times \Pr\{S_t = m;\ Y_t;\ Y_{t+1}^\tau \mid S_{t-1} = m';\ Y_1^{t-1}\} \\ &= \alpha_{t-1}(m') \\ &\quad \times \Pr\{Y_{t+1}^\tau \mid S_{t-1} = m';\ Y_1^{t-1};\ S_t = m;\ Y_t\} \\ &\quad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m';\ Y_1^{t-1}\} \\ &= \alpha_{t-1}(m') \Pr\{Y_{t+1}^\tau \mid S_t = m\} \\ &\quad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\ &= \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m), \end{aligned} \tag{9}$$

and if $t = \tau$, we obtain

$$\begin{aligned} \sigma_t(m', m) &= \Pr\{S_{t-1} = m';\ S_t = m;\ Y_1^\tau\} \\ &= \Pr\{S_{t-1} = m';\ Y_1^{\tau-1};\ S_t = m;\ Y_\tau\} \\ &= \Pr\{S_{t-1} = m';\ Y_1^{t-1};\ S_t = m;\ Y_t\} \\ &= \Pr\{S_{t-1} = m';\ Y_1^{t-1}\} \\ &\quad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m';\ Y_1^{t-1}\} \\ &= \alpha_{t-1}(m') \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m';\ Y_1^{t-1}\} \\ &= \alpha_{t-1}(m') \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\ &= \alpha_{t-1}(m')\gamma_t(m', m) \\ &= \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m). \end{aligned} \tag{10}$$

Here we used the Markov chain property and the definition that $\beta_\tau(m) = 1$.

It remains to check the case $t = 1$. If $t = 1$, we have

$$\begin{aligned}
\sigma_t(m', m) &= \Pr\{S_{t-1} = m';\ S_t = m;\ Y_1^\tau\} \\
&= \Pr\{S_{t-1} = m';\ S_t = m;\ Y_t;\ Y_{t+1}^\tau\} \\
&= \Pr\{S_t = m;\ Y_t;\ Y_{t+1}^\tau \mid S_{t-1} = m'\} \\
&\quad \times \Pr\{S_{t-1} = m'\} \\
&= \Pr\{Y_{t+1}^\tau \mid S_t = m;\ Y_t;\ S_{t-1} = m'\} \\
&\quad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\
&\quad \times \Pr\{S_{t-1} = m'\} \\
&= \Pr\{Y_{t+1}^\tau \mid S_t = m\}\gamma_t(m', m)\Pr\{S_{t-1} = m'\} \\
&= \beta_t(m)\gamma_t(m', m)\alpha_{t-1}(m'),
\end{aligned} \tag{11}$$

where we have defined $\alpha_0(m') = \Pr\{S_{t-1} = m'\}$. Since it is assumed that the recursive systematic convolutional (RSC) code always starts from the all-zero state $S_0 = 0$, we have $\alpha_0(0) = 1$ and $\alpha_0(m) = 0$, $1 \le m \le 2^\nu - 1$.

To proceed further, we digress here to introduce some notation. A directed branch on the trellis diagram of a recursive systematic convolutional (RSC) code is completely characterized by the node it emanates from and the node it reaches. In other words, a directed branch on the trellis diagram of an RSC code is identified by an ordered pair of nonnegative integers $(m', m)$, where $0 \le m', m \le 2^\nu - 1$. We remark here that not every ordered pair of integers $(m', m)$ can be used to identify a directed branch. Let $\mathcal{B}_{t,0} = \{(m', m) : S_{t-1} = m',\ d_t = 0,\ S_t = m\}$ and $\mathcal{B}_{t,1} = \{(m', m) : S_{t-1} = m',\ d_t = 1,\ S_t = m\}$. $\mathcal{B}_{t,0}$ (resp., $\mathcal{B}_{t,1}$) represents the set of all the directed branches on the trellis diagram of an RSC code where the $t$th input bit $d_t$ is 0 (resp., 1).

With the above definitions, we are now in a position to present the forward and backward recursions for the $\alpha$ and $\beta$ sequences and the formula for computing the APP sequence $\Lambda(d_t)$.

In fact, if $2 \le t \le \tau$, we have

$$\begin{aligned}
\alpha_t(m) &= \Pr\{S_t = m;\ Y_1^t\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t-1} = m';\ Y_1^{t-1};\ S_t = m;\ Y_t\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t-1} = m';\ Y_1^{t-1}\} \\
&\qquad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m';\ Y_1^{t-1}\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t-1} = m';\ Y_1^{t-1}\} \\
&\qquad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\
&= \sum_{m'=0}^{2^\nu-1} \alpha_{t-1}(m')\gamma_t(m', m),
\end{aligned} \tag{12}$$

and if $t = 1$, we have

$$\begin{aligned}
\alpha_t(m) &= \Pr\{S_t = m;\ Y_1^t\} = \Pr\{S_t = m;\ Y_t\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t-1} = m';\ S_t = m;\ Y_t\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t-1} = m'\} \\
&\qquad \times \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\
&= \sum_{m'=0}^{2^\nu-1} \alpha_{t-1}(m')\gamma_t(m', m).
\end{aligned} \tag{13}$$

Similarly, if $1 \le t \le \tau - 2$, we have

$$\begin{aligned}
\beta_t(m) &= \Pr\{Y_{t+1}^\tau \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t+1} = m';\ Y_{t+1};\ Y_{t+2}^\tau \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{Y_{t+2}^\tau \mid S_{t+1} = m';\ Y_{t+1};\ S_t = m\} \\
&\qquad \times \Pr\{S_{t+1} = m';\ Y_{t+1} \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{Y_{t+2}^\tau \mid S_{t+1} = m'\} \\
&\qquad \times \Pr\{S_{t+1} = m';\ Y_{t+1} \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_{t+1}(m, m'),
\end{aligned} \tag{14}$$

where we used the Markov chain property of the RSC code. If $t = \tau - 1$, we have

$$\begin{aligned}
\beta_t(m) &= \Pr\{Y_{t+1}^\tau \mid S_t = m\} \\
&= \Pr\{Y_{t+1} \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \Pr\{S_{t+1} = m';\ Y_{t+1} \mid S_t = m\} \\
&= \sum_{m'=0}^{2^\nu-1} \gamma_{t+1}(m, m') \\
&= \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_{t+1}(m, m'),
\end{aligned} \tag{15}$$

where we used the definition that $\beta_\tau(m') = 1$.

We can also easily verify that for $i = 0, 1$,

$$\begin{aligned}
\Pr\{d_t = i \mid Y_1^\tau\} &= \frac{\sum_{(m',m)\in\mathcal{B}_{t,i}} \Pr\{S_{t-1} = m',\ S_t = m,\ Y_1^\tau\}}{\Pr(Y_1^\tau)} \\
&= \sum_{(m',m)\in\mathcal{B}_{t,i}} \frac{\sigma_t(m', m)}{\Pr(Y_1^\tau)}.
\end{aligned} \tag{16}$$

It follows from (2), (9), (10), (11), and (16) that the APP sequence $\Lambda(d_t)$ is computed by

$$\begin{aligned}\Lambda(d_t) &= \log \frac{\sum_{(m',m)\in\mathcal{B}_{t,1}}\left(\sigma_t(m',m)/\Pr\left(Y_1^\tau\right)\right)}{\sum_{(m',m)\in\mathcal{B}_{t,0}}\left(\sigma_t(m',m)/\Pr\left(Y_1^\tau\right)\right)} \\ &= \log \frac{\sum_{(m',m)\in\mathcal{B}_{t,1}}\sigma_t(m',m)}{\sum_{(m',m)\in\mathcal{B}_{t,0}}\sigma_t(m',m)} \\ &= \log \frac{\sum_{(m',m)\in\mathcal{B}_{t,1}}\alpha_{t-1}(m')\gamma_t(m',m)\beta_t(m)}{\sum_{(m',m)\in\mathcal{B}_{t,0}}\alpha_{t-1}(m')\gamma_t(m',m)\beta_t(m)},\end{aligned} \tag{17}$$

where $1 \le t \le \tau$ and $\alpha_{t-1}(m')$ are computed by the forward recursions (12) and (13) and $\beta_t(m)$ are computed by the backward recursions (14) and (15).

Equations (12), (13), (14), (15), and (17) constitute the well-known BCJR MAP algorithm for recursive systematic convolutional codes.

We can further simplify and reformulate the BCJR MAP algorithm for a binary rate $1/n$ recursive systematic convolutional code. In fact,

$$\begin{aligned}\gamma_t(m',m) &= \Pr\{S_t = m;\ Y_t \mid S_{t-1} = m'\} \\ &= \sum_{i=0}^{1}\Pr\{Y_t;\ d_t = i;\ S_t = m \mid S_{t-1} = m'\} \\ &= \sum_{i=0}^{1}\gamma_i(Y_t,m',m),\end{aligned} \tag{18}$$

where

$$\begin{aligned}\gamma_i(Y_t,m',m) &= \Pr\{Y_t;\ d_t = i;\ S_t = m \mid S_{t-1} = m'\} \\ &= \Pr\{Y_t \mid d_t = i;\ S_t = m;\ S_{t-1} = m'\} \\ &\quad \times \Pr\{d_t = i;\ S_t = m \mid S_{t-1} = m'\} \\ &= \Pr\{Y_t \mid d_t = i;\ S_t = m;\ S_{t-1} = m'\} \\ &\quad \times \Pr\{S_t = m \mid d_t = i;\ S_{t-1} = m'\} \\ &\quad \times \Pr\{d_t = i \mid S_{t-1} = m'\} \\ &= \Pr\{Y_t \mid d_t = i;\ S_{t-1} = m'\} \\ &\quad \times \Pr\{S_t = m \mid d_t = i;\ S_{t-1} = m'\} \\ &\quad \times \Pr\{d_t = i\}.\end{aligned} \tag{19}$$

Substituting (18), (19) into (12) and (13), we obtain

$$\begin{aligned}\alpha_t(m) &= \sum_{m'=0}^{2^\nu-1}\alpha_{t-1}(m')\gamma_t(m',m) \\ &= \sum_{m'=0}^{2^\nu-1}\alpha_{t-1}(m')\sum_{j=0}^{1}\gamma_j(Y_t,m',m) \\ &= \sum_{m'=0}^{2^\nu-1}\alpha_{t-1}(m') \\ &\quad \times \sum_{j=0}^{1}\Pr\{Y_t \mid d_t = j;\ S_{t-1} = m'\} \\ &\qquad \times \Pr\{S_t = m \mid d_t = j;\ S_{t-1} = m'\} \\ &\qquad \times \Pr\{d_t = j\} \\ &= \sum_{j=0}^{1}\alpha_{t-1}(S_b^j(m))\Pr\{d_t = j\} \\ &\qquad \times \Pr\{Y_t \mid d_t = j;\ S_{t-1} = S_b^j(m)\}.\end{aligned} \tag{20}$$

Here we used the fact that for any given state $m$, the probability $\Pr\{S_t = m \mid d_t = j;\ S_{t-1} = m'\}$ is nonzero if and only if $m' = S_b^j(m)$ and $\Pr\{S_t = m \mid d_t = j;\ S_{t-1} = S_b^j(m)\} = 1$. By Proposition A.1 in the appendix, we have, for $j = 0, 1$,

$$\Pr\{d_t = j\} = \frac{\exp\left(L_a(d_t)j\right)}{1+\exp\left(L_a(d_t)\right)}. \tag{21}$$

By Proposition A.2 in the appendix, we also have, for $j = 0, 1$, and $0 \le m \le 2^\nu - 1$,

$$\begin{aligned}&\Pr\{Y_t \mid d_t = j;\ S_{t-1} = S_b^j(m)\} \\ &\quad = \mu_t \exp\left(L_c r_t^{(1)} j + \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(j, S_b^j(m))\right),\end{aligned} \tag{22}$$

where $\mu_t > 0$ is a positive constant independent of $j$ and $m$ and $L_c = 2/\sigma^2$ is called the channel reliability coefficient. Using (21) and (22), the identity (20) can be rewritten as

$$\begin{aligned}\alpha_t(m) &= \sum_{j=0}^{1}\alpha_{t-1}(S_b^j(m))\Pr\{d_t = j\} \\ &\qquad \times \Pr\{Y_t \mid d_t = j;\ S_{t-1} = S_b^j(m)\} \\ &= \delta_t\sum_{j=0}^{1}\alpha_{t-1}(S_b^j(m)) \\ &\qquad \times \exp j\left(L_a(d_t) + L_c r_t^{(1)}\right) \\ &\qquad \times \exp\sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(j, S_b^j(m)) \\ &= \delta_t\sum_{j=0}^{1}\alpha_{t-1}(S_b^j(m))\Gamma_t(j, S_b^j(m)),\end{aligned} \tag{23}$$

where $\delta_t = \mu_t/(1+\exp(L_a(d_t)))$ and for $j = 0,1$ and $0 \le m \le 2^\nu - 1$, $\Gamma_t(j,m)$ is defined by

$$\Gamma_t(j,m) = \exp\left(j\left(L_a(d_t) + L_c r_t^{(1)}\right) + \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(j,m)\right). \tag{24}$$

Similarly, from (14), (15), (18), (19), and using Propositions A.1 and A.2 in the appendix, it can be shown that

$$\begin{aligned}
\beta_t(m) &= \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_{t+1}(m,m') \\
&= \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m') \sum_{j=0}^{1} \gamma_j(Y_{t+1},m,m') \\
&= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m') \Pr\{d_{t+1}=j\} \\
&\qquad \times \Pr\{Y_{t+1} \mid d_{t+1}=j;\ S_t=m\} \\
&\qquad \times \Pr\{S_{t+1}=m' \mid d_{t+1}=j;\ S_t=m\} \\
&= \sum_{j=0}^{1} \beta_{t+1}(S_f^j(m)) \Pr\{d_{t+1}=j\} \\
&\qquad \times \Pr\{Y_{t+1} \mid d_{t+1}=j;\ S_t=m\} \\
&= \delta_{t+1}\sum_{j=0}^{1} \beta_{t+1}(S_f^j(m)) \\
&\qquad \times \exp j\left(L_a(d_{t+1}) + L_c r_{t+1}^{(1)}\right) \\
&\qquad \times \exp \sum_{p=2}^{n} L_c r_{t+1}^{(p)} Y_{p-1}(j,m) \\
&= \delta_{t+1}\sum_{j=0}^{1} \beta_{t+1}(S_f^j(m))\Gamma_{t+1}(j,m),
\end{aligned} \tag{25}$$

where $\delta_{t+1} = \mu_{t+1}/(1+\exp(L_a(d_{t+1})))$.

Using mathematical induction, it can be shown that the multiplicative constants $\delta_t$, $\delta_{t+1}$ can be set to 1 without changing the APP sequence $\Lambda(d_t)$ (cf. Proposition A.3 in the appendix) and the BCJR MAP algorithm can be finally formulated as follows. Let the $\alpha$ sequence be computed by the forward recursion

$$\begin{gathered}
\alpha_0(0) = 1, \\
\alpha_0(m) = 0, \quad 1 \le m \le 2^\nu - 1, \\
\alpha_t(m) = \sum_{j=0}^{1} \alpha_{t-1}(S_b^j(m))\Gamma_t(j,S_b^j(m)), \\
1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1,
\end{gathered} \tag{26}$$

and let the $\beta$ sequence be computed by the backward recursion

$$\begin{gathered}
\beta_\tau(m) = 1, \quad 0 \le m \le 2^\nu - 1, \\
\beta_t(m) = \sum_{j=0}^{1} \beta_{t+1}(S_f^j(m))\Gamma_{t+1}(j,m), \\
1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1.
\end{gathered} \tag{27}$$

The APP sequence $\Lambda(d_t)$ is then computed by

$$\begin{aligned}
\Lambda(d_t) &= \log \frac{\sum_{(m,m')\in\mathcal{B}_{t,1}} \alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')}{\sum_{(m,m')\in\mathcal{B}_{t,0}} \alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')} \\
&= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\gamma_t(m,S_f^1(m))\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\gamma_t(m,S_f^0(m))\beta_t(S_f^0(m))} \\
&= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\gamma_1(Y_t,m,S_f^1(m))\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\gamma_0(Y_t,m,S_f^0(m))\beta_t(S_f^0(m))} \\
&= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\Gamma_t(1,m)\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\Gamma_t(0,m)\beta_t(S_f^0(m))} \\
&= L_a(d_t) + L_c r_t^{(1)} + \Lambda_e(d_t),
\end{aligned} \tag{28}$$

where $\Lambda_e(d_t)$, the extrinsic information for data bit $d_t$, is defined by

$$\begin{gathered}
\Lambda_e(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\eta_1(m)\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\eta_0(m)\beta_t(S_f^0(m))}, \\
\eta_i(m) = \exp \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(i,m), \quad i = 0,1.
\end{gathered} \tag{29}$$

The BCJR MAP algorithm can be reformulated systematically in a number of different ways, resulting in the so-called modified BCJR MAP algorithms. They are discussed in the following sections.

## 3. THE SBGT MAP ALGORITHM

In this section, we derive the SBGT MAP algorithm from the BCJR MAP algorithm. For $i = 0,1$ and $1 \le t \le \tau$, let

$$\alpha_t^i(m) = \sum_{(m',m)\in\mathcal{B}_{t,i}} \alpha_{t-1}(m')\gamma_t(m',m). \tag{30}$$

Equation (17) can then be rewritten as

$$\Lambda(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_t^1(m)\beta_t(m)}{\sum_{m=0}^{2^\nu-1} \alpha_t^0(m)\beta_t(m)}, \tag{31}$$

since

$$\frac{\sum_{(m',m)\in\mathcal{B}_{t,1}}\alpha_{t-1}(m')\gamma_t(m',m)\beta_t(m)}{\sum_{(m',m)\in\mathcal{B}_{t,0}}\alpha_{t-1}(m')\gamma_t(m',m)\beta_t(m)} = \frac{\sum_{m=0}^{2^\nu-1}\beta_t(m)\sum_{(m',m)\in\mathcal{B}_{t,1}}\alpha_{t-1}(m')\gamma_t(m',m)}{\sum_{m=0}^{2^\nu-1}\beta_t(m)\sum_{(m',m)\in\mathcal{B}_{t,0}}\alpha_{t-1}(m')\gamma_t(m',m)} = \frac{\sum_{m=0}^{2^\nu-1}\alpha_t^1(m)\beta_t(m)}{\sum_{m=0}^{2^\nu-1}\alpha_t^0(m)\beta_t(m)}. \tag{32}$$

Moreover, $\alpha_t^i(m)$ admits the probabilistic interpretation:

$$\begin{aligned}\alpha_t^i(m) &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\alpha_{t-1}(m')\gamma_t(m',m)\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\Pr\{S_{t-1}=m';\ Y_1^{t-1}\}\\ &\qquad\times\Pr\{S_t=m;\ Y_t\mid S_{t-1}=m'\}\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\Pr\{S_{t-1}=m';\ Y_1^{t-1}\}\\ &\qquad\times\Pr\{S_t=m;\ Y_t\mid S_{t-1}=m';\ Y_1^{t-1}\}\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\Pr\{S_t=m;\ Y_1^t;\ S_{t-1}=m'\}\\ &= \Pr\{d_t=i;\ S_t=m;\ Y_1^t\}.\end{aligned} \tag{33}$$

It is shown below that $\alpha_t^i(m)$ can be computed by the following forward recursions

$$\begin{gathered}\alpha_0^0(0)=\alpha_0^1(0)=1,\\ \alpha_0^i(m)=0,\quad 1\le m\le 2^\nu-1,\ i=0,1,\\ \alpha_t^i(m)=\sum_{m'=0}^{2^\nu-1}\sum_{j=0}^{1}\alpha_{t-1}^j(m')\,\gamma_i(Y_t,m',m),\\ 1\le t\le\tau,\quad i=0,1,\quad 0\le m\le 2^\nu-1,\end{gathered} \tag{34}$$

and $\beta_t(m)$ can be computed by (14), (15), and (18), which are repeated here for easy reference:

$$\begin{gathered}\beta_\tau(m)=1,\quad 0\le m\le 2^\nu-1,\\ \beta_t(m)=\sum_{m'=0}^{2^\nu-1}\sum_{i=0}^{1}\beta_{t+1}(m')\gamma_i(Y_{t+1},m,m'),\\ 1\le t\le\tau-1,\quad 0\le m\le 2^\nu-1.\end{gathered} \tag{35}$$

In fact, from (12) and (13), it follows that for $1\le t\le\tau$,

$$\begin{aligned}\alpha_t(m) &= \sum_{m'=0}^{2^\nu-1}\alpha_{t-1}(m')\gamma_t(m',m)\\ &= \sum_{j=0}^{1}\sum_{(m',m)\in\mathcal{B}_{t,j}}\alpha_{t-1}(m')\gamma_t(m',m)\\ &= \sum_{j=0}^{1}\alpha_t^j(m).\end{aligned} \tag{36}$$

Substituting (36) into (30), we obtain, for $2\le t\le\tau$,

$$\begin{aligned}\alpha_t^i(m) &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\alpha_{t-1}(m')\gamma_t(m',m)\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\sum_{j=0}^{1}\alpha_{t-1}^j(m')\gamma_t(m',m)\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\sum_{j=0}^{1}\alpha_{t-1}^j(m')\\ &\qquad\times[\gamma_i(Y_t,m',m)+\gamma_{1-i}(Y_t,m',m)]\\ &= \sum_{(m',m)\in\mathcal{B}_{t,i}}\sum_{j=0}^{1}\alpha_{t-1}^j(m')\gamma_i(Y_t,m',m)\\ &= \sum_{m'=0}^{2^\nu-1}\sum_{j=0}^{1}\alpha_{t-1}^j(m')\gamma_i(Y_t,m',m).\end{aligned} \tag{37}$$

Here we used (18) and the fact that for any $m'$ with $(m',m)\in\mathcal{B}_{t,i}$, $\gamma_{1-i}(Y_t,m',m) = 0$ (cf. Proposition A.4 in the appendix). This proves the forward recursions (34) for $2\le t\le\tau$. Using (30) and the fact that $\alpha_0(0) = 1$ and $\alpha_0(m) = 0$, $m\ne 0$, it can be verified directly that the forward recursion (37) holds also for $t=1$ if $\alpha_0^i(m)$ are defined by

$$\begin{gathered}\alpha_0^0(0)=\alpha_0^1(0)=\frac{1}{2},\\ \alpha_0^i(m)=0,\quad 1\le m\le 2^\nu-1,\ i=0,1.\end{gathered} \tag{38}$$

Using essentially the same argument as the one used in the proof of Proposition A.3 in the appendix, it can be shown that the values of $\alpha_0^i(m)$ can be reinitialized as $\alpha_0^j(0) = 1$, $\alpha_0^j(m') = 0$, $j = 0,1$, $m'\ne 0$. This proves the forward recursions (34) for $\alpha_t^i(m)$.

Equations (34), (35), and (31) constitute a simplified version of the modified BCJR MAP algorithm developed by Berrou et al. in the classical paper [1]. We remark here that the main difference between the version presented here and the version in [1] is that the redundant divisions in [1, equations (20), (21)] are now removed. As mentioned in the introduction, for brevity, the modified BCJR MAP algorithm of [1] is called the BGT MAP algorithm and its simplified version presented in this section is called the SBGT MAP algorithm (or simply called the SBGT algorithm).

Using (19), (A.2), (A.4), and applying a mathematical induction argument similar to the one used in the proof of Proposition A.3 in the appendix, the SBGT MAP algorithm can be further simplified and reformulated. Details are omitted here due to space limitations and the reader is referred to [3] for similar simplifications. In summary, the APP sequence $\Lambda(d_t)$ is computed by (31), where $\alpha_t^i(m)$ are computed by the forward recursions

$$\begin{aligned}
&\alpha_0^0(0) = \alpha_0^1(0) = 1,\\
&\alpha_0^i(m) = 0, \quad 1 \le m \le 2^\nu - 1,\ i = 0,1,\\
&\alpha_t^i(m) = \left(\sum_{j=0}^{1} \alpha_{t-1}^j\big(S_b^i(m)\big)\right)\Gamma_t\big(i, S_b^i(m)\big),\\
&1 \le t \le \tau, \quad i = 0,1, \quad 0 \le m \le 2^\nu - 1,
\end{aligned} \tag{39}$$

and $\beta_t(m)$ are computed by (27) which is repeated here for easy reference and comparisons:

$$\begin{aligned}
&\beta_\tau(m) = 1, \quad 0 \le m \le 2^\nu - 1,\\
&\beta_t(m) = \sum_{j=0}^{1} \beta_{t+1}\big(S_f^j(m)\big)\Gamma_{t+1}(j,m),\\
&1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1.
\end{aligned} \tag{40}$$

Note that the branch metric $\Gamma_t(j,m)$ is defined in (24).

## 4. THE DUAL SBGT (DSBGT) MAP ALGORITHM

This section derives from the BCJR MAP algorithm a dual version of the SBGT MAP algorithm. For $i = 0,1$, and $1 \le t \le \tau$, let

$$\beta_t^i(m) = \sum_{(m,m') \in \mathcal{B}_{t,i}} \gamma_t(m,m')\beta_t(m'). \tag{41}$$

Using this notation, (17) can be rewritten as

$$\Lambda(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^1(m)}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^0(m)}, \quad 1 \le t \le \tau, \tag{42}$$

since

$$\begin{aligned}
&\frac{\sum_{(m,m') \in \mathcal{B}_{t,1}} \alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')}{\sum_{(m,m') \in \mathcal{B}_{t,0}} \alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')}\\
&\quad = \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m) \sum_{(m,m') \in \mathcal{B}_{t,1}} \gamma_t(m,m')\beta_t(m')}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m) \sum_{(m,m') \in \mathcal{B}_{t,0}} \gamma_t(m,m')\beta_t(m')}\\
&\quad = \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^1(m)}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^0(m)}, \quad 1 \le t \le \tau.
\end{aligned} \tag{43}$$

Moreover, $\beta_t^i(m)$ admits the probabilistic interpretation:

$$\begin{aligned}
\beta_t^i(m) &= \sum_{(m,m') \in \mathcal{B}_{t,i}} \gamma_t(m,m')\beta_t(m')\\
&= \sum_{(m,m') \in \mathcal{B}_{t,i}} \Pr\{S_t = m';\ Y_t \mid S_{t-1} = m\}\\
&\qquad \times \Pr\{Y_{t+1}^\tau \mid S_t = m'\}\\
&= \sum_{(m,m') \in \mathcal{B}_{t,i}} \Pr\{S_t = m';\ Y_t \mid S_{t-1} = m\}\\
&\qquad \times \Pr\{Y_{t+1}^\tau \mid S_t = m';\ Y_t\}\\
&= \Pr\{d_t = i;\ Y_t^\tau \mid S_{t-1} = m\}.
\end{aligned} \tag{44}$$

The sequence $\alpha_t(m)$ is computed recursively by (12), (13), and (18), which are repeated here for easy reference and comparisons:

$$\begin{aligned}
&\alpha_0(0) = 1,\\
&\alpha_0(m) = 0, \quad 1 \le m \le 2^\nu - 1,\\
&\alpha_t(m) = \sum_{m'=0}^{2^\nu-1} \sum_{i=0}^{1} \alpha_{t-1}(m')\gamma_i(Y_t, m', m),\\
&1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1.
\end{aligned} \tag{45}$$

The sequence $\beta_t^i(m)$ is computed recursively by the following backward recursions as will be shown next:

$$\begin{aligned}
&\beta_{\tau+1}^i(m) = 1, \quad i = 0,1,\ 0 \le m \le 2^\nu - 1,\\
&\beta_t^i(m) = \sum_{m'=0}^{2^\nu-1} \sum_{j=0}^{1} \beta_{t+1}^j(m')\gamma_i(Y_t, m, m'),\\
&1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1.
\end{aligned} \tag{46}$$

In fact, from (14) and (15), it follows that for $1 \le t \le \tau - 1$,

$$\begin{aligned}
\beta_t(m) &= \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_{t+1}(m,m')\\
&= \sum_{j=0}^{1} \sum_{(m,m') \in \mathcal{B}_{t+1,j}} \beta_{t+1}(m')\gamma_{t+1}(m,m')\\
&= \sum_{j=0}^{1} \beta_{t+1}^j(m).
\end{aligned} \tag{47}$$

Substituting (47) into (41) and using (18), we obtain, for $1 \le t \le \tau - 1$,

$$\begin{aligned}\beta_t^i(m) &= \sum_{(m,m')\in\mathcal{B}_{t,i}} \gamma_t(m,m')\beta_t(m') \\ &= \sum_{(m,m')\in\mathcal{B}_{t,i}} \gamma_t(m,m') \sum_{j=0}^{1} \beta_{t+1}^j(m') \\ &= \sum_{(m,m')\in\mathcal{B}_{t,i}} \left[\gamma_i(Y_t,m,m') + \gamma_{1-i}(Y_t,m,m')\right] \\ &\quad \times \sum_{j=0}^{1} \beta_{t+1}^j(m') \\ &= \sum_{(m,m')\in\mathcal{B}_{t,i}} \sum_{j=0}^{1} \gamma_i(Y_t,m,m')\beta_{t+1}^j(m') \\ &= \sum_{m'=0}^{2^\nu-1} \sum_{j=0}^{1} \gamma_i(Y_t,m,m')\beta_{t+1}^j(m').\end{aligned} \tag{48}$$

Here we used the fact that for $(m,m')\in\mathcal{B}_{t,i}$, $\gamma_{1-i}(Y_t,m,m')=0$ (cf. Proposition A.4 in the appendix). This proves the backward recursions (46) for $1 \le t \le \tau-1$. Using (41) and the fact that $\beta_\tau(m) = 1$, $0 \le m \le 2^\nu - 1$, it can also be verified that (48) holds for $t = \tau$ if $\beta_{\tau+1}^j(m')$ is defined by $\beta_{\tau+1}^j(m') = 1/2$, $0 \le m' \le 2^\nu - 1$.

As in the derivation of the SBGT MAP algorithm, using a mathematical induction argument similar to the one used in the proof of Proposition A.3 in the appendix, it can be shown that the values of $\beta_{\tau+1}^j(m')$ can be reinitialized as $\beta_{\tau+1}^j(m') = 1$, $j = 0,1$, $m' = 0,1,\ldots,2^\nu - 1$, without having any impact on the final computation of $\Lambda(d_t)$. This completes the proof of the backward recursive relations (46) for the $\beta_t^i(m)$ sequence.

Equations (45), (46), and (42) constitute an MAP algorithm that is dual in structure to the SBGT MAP algorithm. It is thus called the dual SBGT MAP algorithm in [3]. In this paper, the dual SBGT MAP algorithm will be called the DSBGT MAP algorithm (or simply called the DSBGT algorithm).

Using (19), (A.2), (A.4), and applying a mathematical induction argument similar to the one used in the proof of Proposition A.3 in the appendix, the DSBGT MAP algorithm can be further simplified and reformulated (details are omitted). The APP sequence $\Lambda(d_t)$ is computed by (42) where $\alpha_t(m)$ are computed by (26) which is repeated here for easy reference and comparisons:

$$\begin{gathered}\alpha_0(0) = 1, \\ \alpha_0(m) = 0, \quad 1 \le m \le 2^\nu - 1, \\ \alpha_t(m) = \sum_{j=0}^{1} \alpha_{t-1}(S_b^j(m))\Gamma_t(j, S_b^j(m)), \\ 1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1,\end{gathered} \tag{49}$$

and $\beta_t^i(m)$ are computed by the backward recursions

$$\begin{gathered}\beta_{\tau+1}^i(m) = 1, \quad 0 \le m \le 2^\nu - 1,\ i = 0,1, \\ \beta_t^i(m) = \left(\sum_{j=0}^{1} \beta_{t+1}^j(S_f^i(m))\right)\Gamma_t(i,m), \\ 1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1, \quad i = 0,1.\end{gathered} \tag{50}$$

## 5. THE PB MAP ALGORITHM DERIVED FROM THE SBGT MAP ALGORITHM

In this section, we show that the modified BCJR MAP algorithm of Pietrobon and Barbulescu can be derived from the SBGT MAP algorithm via simple permutations.

In fact, since the two mappings $S_f^1$ and $S_f^0$ are one-to-one correspondences from the set $\{0,1,2,\ldots,2^\nu - 1\}$ onto itself, from (31) it follows that the APP sequence $\Lambda(d_t)$ can be rewritten as

$$\begin{aligned}\Lambda(d_t) &= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_t^1(m)\beta_t(m)}{\sum_{m=0}^{2^\nu-1} \alpha_t^0(m)\beta_t(m)} \\ &= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_t^1(S_f^1(m))\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_t^0(S_f^0(m))\beta_t(S_f^0(m))}.\end{aligned} \tag{51}$$

Define

$$\begin{aligned} a_t^i(m) &= \alpha_t^i(S_f^i(m)), \\ b_t^i(m) &= \beta_t(S_f^i(m)).\end{aligned} \tag{52}$$

Then the APP sequence $\Lambda(d_t)$ can be computed by

$$\Lambda(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} a_t^1(m)b_t^1(m)}{\sum_{m=0}^{2^\nu-1} a_t^0(m)b_t^0(m)}. \tag{53}$$

It can be verified that

$$\begin{aligned} a_t^i(m) &= \alpha_t^i(S_f^i(m)) \\ &= \Pr\{d_t = i;\ S_t = S_f^i(m);\ Y_1^t\} \\ &= \Pr\{d_t = i;\ S_{t-1} = m;\ Y_1^t\}, \\ &\quad 1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1, \\ b_t^i(m) &= \beta_t(S_f^i(m)) \\ &= \Pr\{Y_{t+1}^\tau \mid S_t = S_f^i(m)\} \\ &= \Pr\{Y_{t+1}^\tau \mid d_t = i;\ S_{t-1} = m\}, \\ &\quad 1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1.\end{aligned} \tag{54}$$

The two equations of (54) show that $a_t^i(m)$ and $b_t^i(m)$ are exactly the same as the $\alpha_t^i(m)$ and $\beta_t^i(m)$ sequences defined in [5].

We can immediately derive the forward and backward recursions for $a_t^i(m)$ and $b_t^i(m)$ from the recursions (34) and (35).

In fact, from the third equation of (34) it follows that for $1 \le t \le \tau$,

$$\begin{aligned} a_t^i(m) &= \alpha_t^i(S_f^i(m)) \quad \text{(by definition)} \\ &= \sum_{m'=0}^{2^\nu-1}\sum_{j=0}^{1} \alpha_{t-1}^j(m')\gamma_i(Y_t, m', S_f^i(m)) \quad \text{(by (34))} \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \alpha_{t-1}^j(m')\gamma_i(Y_t, m', S_f^i(m)) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \alpha_{t-1}^j(S_f^j(m'))\ \gamma_i(Y_t, S_f^j(m'), S_f^i(m)) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} a_{t-1}^j(m')\gamma_i(Y_t, S_f^j(m'), S_f^i(m)) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} a_{t-1}^j(m')\gamma_{j,i}(Y_t, m', m), \end{aligned} \tag{55}$$

where

$$\gamma_{j,i}(Y_t, m', m) = \gamma_i(Y_t, S_f^j(m'), S_f^i(m)). \tag{56}$$

From the first and second equations of (34), it follows that for $i = 0, 1$,

$$\begin{aligned} a_0^i(m) &= 1, \quad \text{for } m = S_b^i(0), \\ a_0^i(m) &= 0, \quad \text{for } m \ne S_b^i(0). \end{aligned} \tag{57}$$

The backward recursions for $b_t^i(m)$ are similarly derived. In fact, from the second equation of (35), it follows that for $1 \le t \le \tau - 1$,

$$\begin{aligned} b_t^i(m) &= \beta_t(S_f^i(m)) \quad \text{(by definition)} \\ &= \sum_{m'=0}^{2^\nu-1}\sum_{j=0}^{1} \beta_{t+1}(m')\gamma_j(Y_{t+1}, S_f^i(m), m') \quad \text{(by (35))} \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_j(Y_{t+1}, S_f^i(m), m') \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \beta_{t+1}(S_f^j(m'))\gamma_j(Y_{t+1}, S_f^i(m), S_f^j(m')) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} b_{t+1}^j(m')\gamma_j(Y_{t+1}, S_f^i(m), S_f^j(m')) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} b_{t+1}^j(m')\gamma_{i,j}(Y_{t+1}, m, m'), \end{aligned} \tag{58}$$

and from the first equation of (35), it follows that

$$b_\tau^i(m) = \beta_\tau(S_f^i(m)) = 1, \quad i = 0, 1. \tag{59}$$

Equations (53), (55), (56), (57), (58), and (59) constitute the modified BCJR MAP algorithm of Pietrobon and Barbulescu developed in [5]. As mentioned in the introduction, for brevity, this algorithm is also called the PB MAP algorithm (or simply called the PB algorithm).

Using (19), (A.2), (A.4), and applying a mathematical induction argument similar to the one used in the proof of Proposition A.3 in the appendix, the PB MAP algorithm can be further simplified and reformulated as follows. The APP sequence $\Lambda(d_t)$ is computed by (53), where $a_t^i(m)$ are computed by the forward recursions

$$\begin{aligned} a_0^i(m) &= 1, \quad m = S_b^i(0), \\ a_0^i(m) &= 0, \quad m \ne S_b^i(0), \\ a_t^i(m) &= \left(\sum_{j=0}^{1} a_{t-1}^j(S_b^j(m))\right)\Gamma_t(i, m), \\ &1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1, \end{aligned} \tag{60}$$

and $b_t^i(m)$ are computed by the backward recursions

$$\begin{aligned} b_\tau^i(m) &= 1, \quad 0 \le m \le 2^\nu - 1, \\ b_t^i(m) &= \sum_{j=0}^{1} b_{t+1}^j(S_f^i(m))\Gamma_{t+1}(j, S_f^i(m)), \\ &1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1. \end{aligned} \tag{61}$$

## 6. THE DUAL PB (DPB) MAP ALGORITHM

The dual SBGT (DSBGT) MAP algorithm presented in Section 4 can be reformulated via permutations to obtain a dual version of the PB MAP algorithm.

In fact, since the two mappings $S_b^1$ and $S_b^0$ are one-to-one correspondences from the set $\{0, 1, 2, \dots, 2^\nu - 1\}$ onto itself, from (42) it follows that the APP sequence $\Lambda(d_t)$ can be rewritten as

$$\begin{aligned} \Lambda(d_t) &= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^1(m)}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\beta_t^0(m)} \\ &= \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(S_b^1(m))\beta_t^1(S_b^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(S_b^0(m))\beta_t^0(S_b^0(m))}. \end{aligned} \tag{62}$$

Define

$$\begin{aligned} g_t^i(m) &= \alpha_{t-1}(S_b^i(m)), \\ h_t^i(m) &= \beta_t^i(S_b^i(m)). \end{aligned} \tag{63}$$

Then the APP sequence $\Lambda(d_t)$ can be computed by

$$\Lambda(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} g_t^1(m)h_t^1(m)}{\sum_{m=0}^{2^\nu-1} g_t^0(m)h_t^0(m)}. \tag{64}$$

The two sequences $g_t^i(m)$ and $h_t^i(m)$ admit the following probabilistic interpretations:

$$\begin{aligned} g_t^i(m) &= \alpha_{t-1}(S_b^i(m)) \\ &= \Pr\{S_{t-1} = S_b^i(m);\ Y_1^{t-1}\}, \quad 2 \le t \le \tau, \\ h_t^i(m) &= \beta_t^i(S_b^i(m)) \\ &= \Pr\{d_t = i;\ Y_t^\tau \mid S_{t-1} = S_b^i(m)\}, \quad 1 \le t \le \tau. \end{aligned} \tag{65}$$

From the third equation of (45), it follows that for $2 \le t \le \tau$,

$$\begin{aligned} g_t^i(m) &= \alpha_{t-1}(S_b^i(m)) \quad \text{(by definition)} \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \alpha_{t-2}(m')\gamma_j(Y_{t-1}, m', S_b^i(m)) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \alpha_{t-2}(S_b^j(m'))\gamma_j(Y_{t-1}, S_b^j(m'), S_b^i(m)) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} g_{t-1}^j(m')\gamma_j(Y_{t-1}, S_b^j(m'), S_b^i(m)), \end{aligned} \tag{66}$$

and from the first and second equations of (45), we obtain

$$\begin{aligned} g_1^i(m) &= 1, \quad \text{if } m = S_f^i(0), \\ g_1^i(m) &= 0, \quad \text{if } m \ne S_f^i(0). \end{aligned} \tag{67}$$

Similarly, from the second equation of (46), it follows that for $1 \le t \le \tau$,

$$\begin{aligned} h_t^i(m) &= \beta_t^i(S_b^i(m)) \quad \text{(by definition)} \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \beta_{t+1}^j(m')\gamma_i(Y_t, S_b^i(m), m') \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} \beta_{t+1}^j(S_b^j(m'))\gamma_i(Y_t, S_b^i(m), S_b^j(m')) \\ &= \sum_{j=0}^{1}\sum_{m'=0}^{2^\nu-1} h_{t+1}^j(m')\gamma_i(Y_t, S_b^i(m), S_b^j(m')), \end{aligned} \tag{68}$$

and from the first equation of (46), it follows that

$$h_{\tau+1}^i(m) = 1, \quad i = 0, 1. \tag{69}$$

Equations (64), (66), (67), (68), and (69) constitute a dual version of the modified BCJR MAP algorithm of Pietrobon and Barbulescu. For brevity, it is called the dual PB (DPB) MAP algorithm (or simply called the DPB algorithm). The duality that exists between the PB MAP algorithm and the DPB MAP algorithm derives from the fact that the DPB MAP algorithm is obtained by permuting nodes on the trellis diagram of the systematic convolutional code from the DSBGT MAP algorithm while the PB MAP algorithm is obtained in a similar way from the SBGT MAP algorithm.

Using (19), (A.2), (A.4), and applying a mathematical induction argument similar to the one used in the proof of Proposition A.3 in the appendix, the DPB MAP algorithm can be further simplified and reformulated as follows. The APP sequence $\Lambda(d_t)$ is computed by (64), where $g_t^i(m)$ are computed by the forward recursions

$$\begin{aligned} g_1^i(m) &= 1, \quad m = S_f^i(0), \\ g_1^i(m) &= 0, \quad m \ne S_f^i(0), \\ g_t^i(m) &= \sum_{j=0}^{1} g_{t-1}^j(S_b^i(m))\Gamma_{t-1}(j, S_b^j(S_b^i(m))), \\ &2 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1, \end{aligned} \tag{70}$$

and $h_t^i(m)$ are computed by the backward recursions

$$\begin{aligned} h_{\tau+1}^i(m) &= 1, \quad 0 \le m \le 2^\nu - 1, \\ h_t^i(m) &= \left(\sum_{j=0}^{1} h_{t+1}^j(S_f^j(m))\right)\Gamma_t(i, S_b^i(m)), \\ &1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1. \end{aligned} \tag{71}$$

Note that $\Gamma_t(j, m)$ is defined in (24).

## 7. COMPLEXITY AND INITIALIZATION ISSUES

### *7.1. Complexity comparisons in the linear domain*

Dualities between the SBGT and DSBGT and between the PB and DPB MAP algorithms immediately imply that the SBGT and DSBGT MAP algorithms have identical computational complexities and memory requirements and so do the PB and DPB MAP algorithms. We next show that the SBGT and PB MAP algorithms have identical computational complexities and memory requirements too. In fact, for $i = 0, 1$, from the second equation of (61), we obtain

$$\begin{aligned} b_t^i(m) &= \sum_{j=0}^{1} b_{t+1}^j(S_f^i(m))\Gamma_{t+1}(j, S_f^i(m)), \\ &1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1. \end{aligned} \tag{72}$$

Since $S_b^i(S_f^i(m)) = S_f^i(S_b^i(m)) = m$, it follows from (72) that for $1 \le t \le \tau - 1$,

$$\begin{aligned} b_t^{1-i}(m) &= \sum_{j=0}^{1} b_{t+1}^j(S_f^{1-i}(m))\Gamma_{t+1}(j, S_f^{1-i}(m)) \\ &= \sum_{j=0}^{1} b_{t+1}^j(S_f^i(S_b^i(S_f^{1-i}(m)))) \\ &\quad \times \Gamma_{t+1}(j, S_f^i(S_b^i(S_f^{1-i}(m)))) \\ &= \sum_{j=0}^{1} b_{t+1}^j(S_f^i(m'))\Gamma_{t+1}(j, S_f^i(m')) \\ &= b_t^i(m'), \quad m' = S_b^i(S_f^{1-i}(m)). \end{aligned} \tag{73}$$

The identity (73) shows that for any given $t$, $1 \leq t \leq \tau$, the sequence $b_t^0(m)$, $0 \leq m \leq 2^\nu - 1$, can be obtained from the sequence $b_t^1(m')$, $0 \leq m' \leq 2^\nu - 1$, via the permutation $m' = S_b^1(S_f^0(m))$. Thus in the PB MAP algorithm, only one of the two sequences $b_t^1(m)$, $b_t^0(m)$, $1 \leq t \leq \tau$, needs to be computed in the backward recursion. Comparing (39), (40), (60), (61), we see that the SBGT and PB MAP algorithms have identical computational complexities and memory requirements. It follows that the SBGT, DSBGT, PB, and DPB MAP algorithms all have identical computational complexities and memory requirements. To compare the BCJR and the modified BCJR MAP algorithms, it suffices to analyze the BCJR and the DSBGT. We will show next that the BCJR and DSBGT MAP algorithms also have identical computational complexities and memory requirements.

First, we note that the branch metrics $\Gamma_t(i, m)$ are used in both the forward and backward recursions for the BCJR and DSBGT MAP algorithms. To minimize the computational load, the branch metrics $\Gamma_t(i, m)$ are stored and reused (see [7]). Let us first compute the number of arithmetic operations required to decode a single bit for the BCJR. The branch metric $\Gamma_t(j, m)$, defined by (24), is computed by

$$\Gamma_t(j, m) = \exp\left(j\left(L_a(d_t) + L_c r_t^{(1)}\right) + \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(j, m)\right). \tag{74}$$

For each decoded bit, there are a total of $B = \min\{2^{\nu+1}, 2^n\}$ different branch metrics to be calculated, each requiring $n-1$ additions and a single exponentiation. Note that the scaling operation by $L_c$ is performed prior to turbo decoding and therefore should be ignored here. To compute $\alpha_t(m)$ in the forward recursion (26), which is reproduced here,

$$\alpha_t(m) = \sum_{j=0}^{1} \alpha_{t-1}(S_b^j(m))\Gamma_t(j, S_b^j(m)), \tag{75}$$

two multiplications and one addition are needed (assuming that the $B$ branch metrics are already computed and stored). The branch metrics are reused in the backward recursion (27), which is reproduced here,

$$\beta_t(m) = \sum_{j=0}^{1} \beta_{t+1}(S_f^j(m))\Gamma_{t+1}(j, m), \tag{76}$$

hence only two multiplications and one addition are required to compute a single $\beta_t(m)$. Finally, $\Lambda(d_t)$, defined in (28), is computed by

$$\Lambda(d_t) = \log \frac{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\Gamma_t(1, m)\beta_t(S_f^1(m))}{\sum_{m=0}^{2^\nu-1} \alpha_{t-1}(m)\Gamma_t(0, m)\beta_t(S_f^0(m))}. \tag{77}$$

We note that the terms

$$\Gamma_t(0, m)\beta_t(S_f^0(m)), \qquad \Gamma_t(1, m)\beta_t(S_f^1(m)) \tag{78}$$

appear in both the computation of $\beta_{t-1}(m)$ and that of $\Lambda_t(d_t)$. Therefore, $\beta_{t-1}(m)$ and $\Lambda(d_t)$ should be computed at the same time, with the terms in (78) computed only once. It follows that to compute $\Lambda(d_t)$, $2M + 1$ multiplications and $2(M - 1)$ additions are required (the single division is considered equivalent to a multiplication, the single natural logarithm operation is ignored, and $M = 2^\nu$). In total, to decode a single bit, there are $B$ exponentiations, $1+(n-1)B+2M+2(M-1) = (n-1)B+4M-1$ additions, and $2M+2M+2M+1 = 6M+1$ multiplications. To decode a bit, memory is required for $M$ values of $\alpha_t(m)$ and $B$ values of branch metrics $\Gamma_t(j, m)$, resulting in a total of $B+M$ units of memory.

For the DSBGT MAP algorithm, we note that the identity

$$\begin{aligned}\sum_{j=0}^{1} \beta_{t+1}^j(S_f^i(m)) &= \sum_{j=0}^{1} \beta_{t+1}^j(S_f^{1-i}(S_b^{1-i}(S_f^i(m))))\\ &= \sum_{j=0}^{1} \beta_{t+1}^j(S_f^{1-i}(m')), \quad m' = S_b^{1-i}(S_f^i(m))\end{aligned} \tag{79}$$

can be used to reduce the number of additions by half in the computation of $\beta_t^0(m)$ and $\beta_t^1(m)$ (cf. (50)). An examination of (42) and the recursions (49) and (50) then shows that to decode a single bit, there are $B$ exponentiations, $1+(n-1)B+2M+2(M-1) = (n-1)B+4M-1$ additions, and $2M+2M+2M+1 = 6M+1$ multiplications. Memory is required for $M$ values of $\alpha_t(m)$ and $B$ values of branch metrics $\Gamma_t(j, m)$ or a total of $B+M$ units.

The preceding calculations show that indeed the BCJR and DSBGT MAP algorithms have identical computational complexities and memory requirements, and therefore the BCJR and the four modified BCJR MAP algorithms all have identical computational complexities and memory requirements.

### 7.2. Complexity comparisons in the log domain

In the log domain, exponentiation operations in the linear domain disappear, multiplications are converted into additions, and additions in the recursions are converted into the so-called $E$ operation defined in [7, equation (21)] based on the formula

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|x-y|}), \tag{80}$$

where the function $\ln(1 + e^{-|x-y|})$ is replaced by a lookup table. Following the same analysis as in the previous subsection, it can be shown that the BCJR and the four modified BCJR MAP algorithms also have identical computational complexities and memory requirements in the log domain.

### 7.3. Initialization of the backward recursion

In this paper, the BCJR and the four modified BCJR MAP algorithms are formulated for a truncated or nonterminated binary convolutional code. If the binary convolutional code is terminated so that the final encoder state is the zero state, then it can be shown that the $\beta_t(m)$ sequence for the BCJR

MAP algorithm can be initialized by setting $\beta_\tau(0) = 1$ and $\beta_\tau(m) = 0$, $m \neq 0$. To show why this is the case, let us look at the backward recursion (15), which is reproduced here:

$$\beta_t(m) = \sum_{m'=0}^{2^\nu-1} \beta_{t+1}(m')\gamma_{t+1}(m,m'). \tag{81}$$

Since the code is terminated at the zero state, for $t = \tau - 1$, from (18) and (19), we can see that the terms $\gamma_{t+1}(m,m') = \gamma_\tau(m,m')$ in (81) are all zero except for $\gamma_{t+1}(m,0)$, which is the only term that may be nonzero. This implies that the $\beta_t(m)$ sequence can be initialized by setting $\beta_\tau(0) = 1$ and resetting $\beta_\tau(m) = 0$, $1 \le m \le 2^\nu - 1 = M - 1$.

A similar argument applies to the SBGT, DSBGT, PB, and DPB MAP algorithms as well. For a terminated binary convolutional code, we have the following initialization strategies. For the backward recursion (40) in the SBGT MAP algorithm, the sequence $\beta_t(m)$ is initialized by setting $\beta_\tau(0) = 1$ and $\beta_\tau(m) = 0$, $m \neq 0$. For the backward recursions (50) in the DSBGT MAP algorithm, the two sequences $\beta_t^0(m)$ and $\beta_t^1(m)$ can be initialized by setting $\beta_{\tau+1}^0(0) = \beta_{\tau+1}^1(0) = 1$ and $\beta_{\tau+1}^0(m) = \beta_{\tau+1}^1(m) = 0$, $m \neq 0$. For the backward recursions (61) in the PB MAP algorithm, the two sequences are initialized by setting $b_\tau^0(S_b^0(0)) = b_\tau^1(S_b^1(0)) = 1$, $b_\tau^0(m) = 0$, $m \neq S_b^0(0)$ and $b_\tau^1(m) = 0$, $m \neq S_b^1(0)$. Finally, for the backward recursions (71) in the DPB MAP algorithm, the two sequences $h_t^0(m)$ and $h_t^1(m)$ are initialized by setting $h_{\tau+1}^0(S_f^0(0)) = h_{\tau+1}^1(S_f^1(0)) = 1$, $h_{\tau+1}^0(m) = 0$, $m \neq S_f^0(0)$, and $h_{\tau+1}^1(m) = 0$, $m \neq S_f^1(0)$.

## 8. SIMULATIONS

The BCJR and the four modified BCJR MAP algorithms formulated in this paper are all mathematically equivalent and should produce identical results in the linear domain. To verify this, the rate 1/2 and rate 1/3 turbo codes defined in the CDMA2000 standard were tested for the AWGN channel with the interleaver size selected to be 1146. At least, 500 bit errors were accumulated for each selected value of $E_b/N_0$. Under the same simulation conditions (same random number generators starting at the same seeds), it turns out that indeed the BCJR and the four modified BCJR MAP algorithms all have identical BER (bit error rate) and FER (frame error rate) performance. More specifically, they generate exactly the same number of bit errors and exactly the same number of frame errors under identical simulation conditions (cf. Figures 2 and 3).

The BCJR and the four modified BCJR MAP algorithms are expected to have identical performance in the log domain since they have identical performance in the linear domain.

## 9. CONCLUSIONS

In this paper, four different modified BCJR MAP algorithms have been systematically derived from the BCJR MAP algorithm via mathematical transformations. The simple connections among these algorithms are thus established. It is shown that the BCJR and the four modified BCJR MAP algorithms have identical computational complexities and memory requirements. Computer simulations confirmed that the BCJR and the four modified BCJR MAP algorithms all have identical performance in an AWGN channel.

The BCJR and the modified BCJR MAP algorithms presented in this paper are formulated for a rate $1/n$ convolutional code. It can be shown that these algorithms can all be extended to a general rate $k/n$ recursive systematic convolutional code. These extensions will be treated elsewhere.

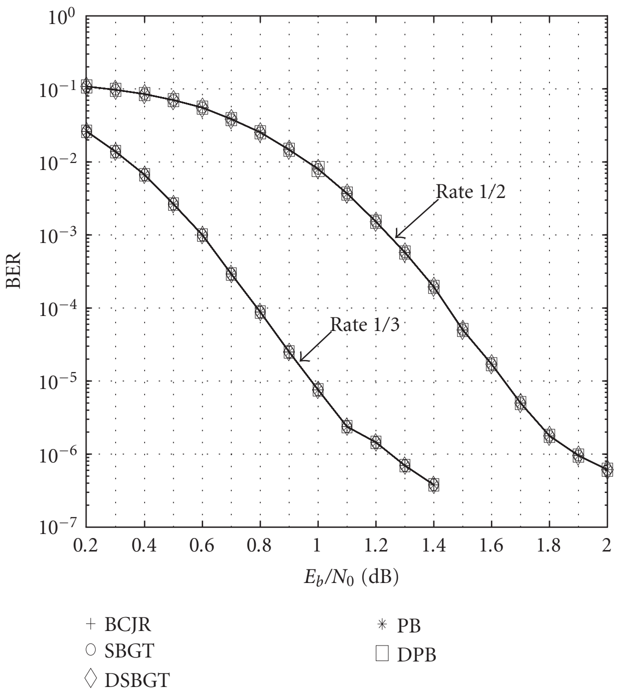

FIGURE 2: BER performance of the rate 1/2 and rate 1/3 turbo codes in CDMA2000 using BCJR and the modified BCJR MAP algorithms (interleaver size = 1146).

## APPENDIX

**Proposition A.1.** *Let*

$$L_a(d_t) = \log \frac{\Pr\{d_t = 1\}}{\Pr\{d_t = 0\}}, \quad 1 \le t \le \tau. \tag{A.1}$$

*For* $i = 0, 1$ *and* $1 \le t \le \tau$, *there exists*

$$\Pr\{d_t = i\} = \frac{\exp(iL_a(d_t))}{1 + \exp L_a(d_t)}. \tag{A.2}$$

*Proof.* It follows from the definition (A.1) and the identity $\Pr\{d_t = 0\} + \Pr\{d_t = 1\} = 1$ that

$$\exp L_a(d_t) = \frac{\Pr\{d_t = 1\}}{\Pr\{d_t = 0\}} = \frac{1}{\Pr\{d_t = 0\}} - 1. \tag{A.3}$$

This implies that $\Pr\{d_t = 0\} = 1/(1 + \exp L_a(d_t))$ and $\Pr\{d_t = 1\} = 1 - \Pr\{d_t = 0\} = \exp L_a(d_t)/(1 + \exp L_a(d_t))$. These two identities combined yield the identity (A.2). □

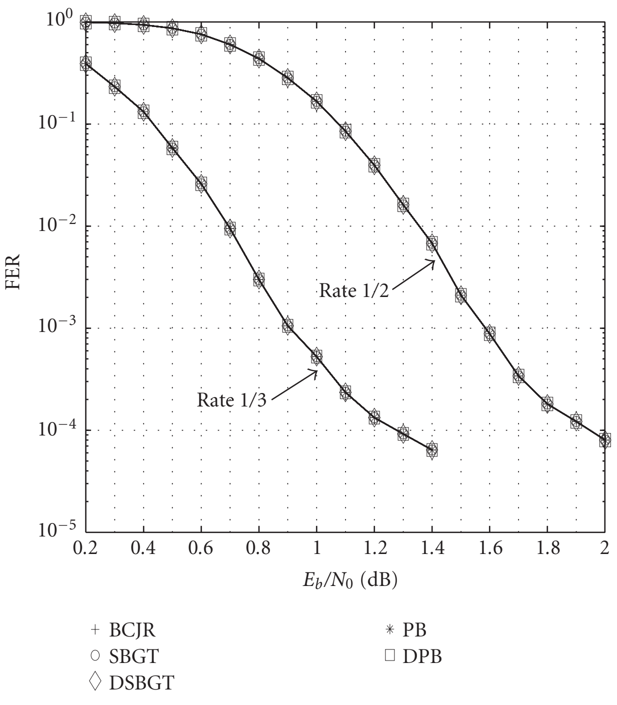

FIGURE 3: FER performance of the rate 1/2 and rate 1/3 turbo codes in CDMA2000 using BCJR and the modified BCJR MAP algorithms (interleaver size = 1146).

**Proposition A.2.** *Let $L_c = 2/\sigma^2$. There exists a positive constant $\mu_t > 0$ such that for $1 \le t \le \tau$, $0 \le m' \le 2^\nu - 1$, and $i = 0, 1$,*

$$\Pr\{Y_t \mid d_t = i;\ S_{t-1} = m'\} = \mu_t \times \exp\Big(L_c r_t^{(1)} i + \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(i, m')\Big), \tag{A.4}$$

*where $\mu_t$ is independent of $i$ and $m'$.*

*Proof.* Let $C_n = (1/\sqrt{2\pi}\sigma)^n$, then

$$\begin{aligned}
\Pr\{Y_t \mid d_t = i;\ S_{t-1} = m'\}
&= C_n \exp\Big(-\frac{1}{2\sigma^2}\big(r_t^{(1)} - (2i-1)\big)^2\Big) \\
&\quad \times \exp\Big(-\frac{1}{2\sigma^2}\sum_{p=1}^{n-1}\Big(r_t^{(p+1)} - \big(2Y_p(i,m') - 1\big)\Big)^2\Big) \\
&= C_n \exp\Big(-\frac{1}{2\sigma^2}\sum_{p=1}^{n}\Big(\big(r_t^{(p)}\big)^2 + 2r_t^{(p)} + 1\Big)\Big) \\
&\quad \times \exp\Big(\frac{2}{\sigma^2} r_t^{(1)} i + \frac{2}{\sigma^2}\sum_{p=2}^{n} r_t^{(p)} Y_{p-1}(i,m')\Big) \\
&= \mu_t \exp\Big(L_c r_t^{(1)} i + \sum_{p=2}^{n} L_c r_t^{(p)} Y_{p-1}(i,m')\Big),
\end{aligned} \tag{A.5}$$

where

$$\mu_t = C_n \exp\Big(-\frac{1}{2\sigma^2}\sum_{p=1}^{n}\Big(\big(r_t^{(p)}\big)^2 + 2r_t^{(p)} + 1\Big)\Big) \tag{A.6}$$

is a positive constant independent of the transmitted data bit $d_t = i$ and $m'$ and $L_c = 2/\sigma^2$. □

**Proposition A.3.** *Let $\alpha_t(m)$, $\beta_t(m)$, and $\Lambda(d_t)$ be defined by* (26), (27), *and* (28), *respectively. Let $\eta_t > 0$ ( $1 \le t \le \tau$) and $\kappa_t > 0$ ( $1 \le t \le \tau - 1$) be two arbitrary sequences of positive constants and define $\bar{\alpha}_t(m)$, $\bar{\beta}_t(m)$, and $\bar{\Lambda}(d_t)$ by*

$$\begin{aligned}
&\bar{\alpha}_0(0) = 1, \\
&\bar{\alpha}_0(m) = 0, \quad 1 \le m \le 2^\nu - 1, \\
&\bar{\alpha}_t(m) = \eta_t \sum_{j=0}^{1} \bar{\alpha}_{t-1}\big(S_b^j(m)\big)\Gamma_t\big(j, S_b^j(m)\big), \\
&1 \le t \le \tau, \quad 0 \le m \le 2^\nu - 1,
\end{aligned} \tag{A.7}$$

$$\begin{aligned}
&\bar{\beta}_\tau(m) = 1, \quad 0 \le m \le 2^\nu - 1, \\
&\bar{\beta}_t(m) = \kappa_t \sum_{j=0}^{1} \bar{\beta}_{t+1}\big(S_f^j(m)\big)\Gamma_{t+1}(j, m), \\
&1 \le t \le \tau - 1, \quad 0 \le m \le 2^\nu - 1,
\end{aligned} \tag{A.8}$$

$$\bar{\Lambda}(d_t) = \log \frac{\sum_{(m,m')\in\mathcal{B}_{t,1}} \bar{\alpha}_{t-1}(m)\gamma_t(m,m')\bar{\beta}_t(m')}{\sum_{(m,m')\in\mathcal{B}_{t,0}} \bar{\alpha}_{t-1}(m)\gamma_t(m,m')\bar{\beta}_t(m')}. \tag{A.9}$$

*It holds that $\Lambda(d_t) = \bar{\Lambda}(d_t)$, $1 \le t \le \tau$.*

*Proof.* We first show by mathematical induction that

$$\bar{\alpha}_t(m) = \eta_0\eta_1\eta_2\cdots\eta_t\alpha_t(m), \quad 0 \le t \le \tau, \tag{A.10}$$

where $\eta_0 = 1$. In fact, (A.10) holds for $t = 0$ since $\bar{\alpha}_0(m) = \alpha_0(m) = \eta_0\alpha_0(m)$, $0 \le m \le 2^\nu - 1$. Next, assume that the identity (A.10) holds for some $t < \tau$. From the third equation of (A.7), it follows that

$$\begin{aligned}
\bar{\alpha}_{t+1}(m) &= \eta_{t+1} \sum_{j=0}^{1} \bar{\alpha}_t\big(S_b^j(m)\big)\Gamma_{t+1}\big(j, S_b^j(m)\big) \\
&= \eta_{t+1} \sum_{j=0}^{1} \eta_0\eta_1\cdots\eta_t\alpha_t\big(S_b^j(m)\big)\Gamma_{t+1}\big(j, S_b^j(m)\big) \\
&= \eta_0\eta_1\cdots\eta_t\eta_{t+1} \sum_{j=0}^{1} \alpha_t\big(S_b^j(m)\big)\Gamma_{t+1}\big(j, S_b^j(m)\big) \\
&= \eta_0\eta_1\cdots\eta_t\eta_{t+1}\alpha_{t+1}(m).
\end{aligned} \tag{A.11}$$

Here we used the assumption that (A.10) holds for $t$ and the third identity of (26). This implies that (A.10) also holds for $t + 1$. By the principle of mathematical induction, (A.10) holds for $0 \le t \le \tau$. This completes the proof of (A.10). In a completely analogous fashion, it can be shown by mathematical induction that

$$\bar{\beta}_t(m) = \kappa_t\kappa_{t+1}\cdots\kappa_\tau\beta_t(m), \quad 1 \le t \le \tau, \tag{A.12}$$

where $\kappa_\tau = 1$. Substituting (A.10) and (A.12) into (A.9), we obtain, for $1 \le t \le \tau$,

$$\begin{aligned}\bar{\Lambda}(d_t) &= \log \frac{\sum_{(m,m')\in\mathcal{B}_{t,1}} \bar{\alpha}_{t-1}(m)\gamma_t(m,m')\bar{\beta}_t(m')}{\sum_{(m,m')\in\mathcal{B}_{t,0}} \bar{\alpha}_{t-1}(m)\gamma_t(m,m')\bar{\beta}_t(m')} \\ &\quad\times \log \frac{\sum_{(m,m')\in\mathcal{B}_{t,1}} C_t\alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')}{\sum_{(m,m')\in\mathcal{B}_{t,0}} C_t\alpha_{t-1}(m)\gamma_t(m,m')\beta_t(m')} \\ &= \Lambda(d_t),\end{aligned} \tag{A.13}$$

where $C_t = \eta_0\eta_1\eta_2\cdots\eta_{t-1}\kappa_t\kappa_{t+1}\cdots\kappa_\tau$. This completes the proof. □

**Proposition A.4.** *For any* $(m', m) \in \mathcal{B}_{t,i}$, $\gamma_{1-i}(Y_t, m', m) = 0$.

*Proof.* Since $(m', m) \in \mathcal{B}_{t,i}$, we have $m = S_f^i(m')$. From (19), it follows that

$$\begin{aligned}\gamma_{1-i}(Y_t, m', m) &= \Pr\{Y_t \mid d_t = 1-i;\ S_{t-1} = m'\} \\ &\quad\times \Pr\{S_t = m \mid d_t = 1-i;\ S_{t-1} = m'\} \\ &\quad\times \Pr\{d_t = 1-i\} \\ &= \Pr\{Y_t \mid d_t = 1-i;\ S_{t-1} = m'\} \\ &\quad\times \Pr\{S_t = S_f^i(m') \mid d_t = 1-i;\ S_{t-1} = m'\} \\ &\quad\times \Pr\{d_t = 1-i\} = 0,\end{aligned} \tag{A.14}$$

since $\Pr\{S_t = S_f^i(m') \mid d_t = 1-i;\ S_{t-1} = m'\} = 0$. In other words, starting from the state $S_{t-1} = m'$ and encoding the bit $1-i$, the new state $S_t$ of the systematic convolutional code will be $S_f^{1-i}(m')$, which is different from $S_f^i(m')$. □

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for comments which led to much improvement of the presentation of this paper. We are especially grateful to one of the reviewers who helped us clarify the complexity issues. We also thank Dr. Mike Sablatash and Dr. John Lodge at the Communications Research Centre (CRC) for reviewing an earlier version of this paper.

## REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes. (1)," in *Proceedings of IEEE International Conference on Communications (ICC '93)*, vol. 2, pp. 1064–1070, Geneva, Switzerland, May 1993.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[3] S. Wang and F. Patenaude, "A simplified BGT MAP algorithm and its dual," in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '03)*, vol. 2, pp. 954–959, Victoria, BC, Canada, August 2003.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[5] S. S. Pietrobon and A. S. Barbulescu, "A simplification of the modified Bahl decoding algorithm for systematic convolutional codes," in *Proceedings of International Symposium on Information Theory & Its Applications (ISITA '94)*, vol. 2, pp. 1073–1077, Sydney, Australia, November 1994.

[6] L. Kleinrock, *Queuing Systems, Volume 1: Theory*, John Wiley & Sons, New York, NY, USA, 1975.

[7] S. S. Pietrobon, "Implementation and performance of a turbo/MAP decoder," *International Journal of Satellite Communications*, vol. 16, no. 1, pp. 23–46, 1998.

**Sichun Wang** obtained his B.S. and M.S. degrees in mathematics from Nankai University, Tianjin, China, in 1983 and 1989, respectively. He obtained his Ph.D. degree in mathematics from McMaster University, Hamilton, ON, Canada, in 1996. During the academic year 1996–1997, he worked at the Communications Research Laboratory of McMaster University as a Postdoctoral Fellow. Since September 1997, he has been working in Ottawa, Canada. He was a Research Scientist at Telexis Corporation and Intrinsix Canada, and a Research Consultant for Calian Corporation. Currently, he works with Defence R & D Canada – Ottawa (DRDC Ottawa). His more recent research has focused on FFT filter-bank-based constant false alarm rate (CFAR) detectors and forward error-correction codes.

**François Patenaude** received the B.A.S. degree from the University of Sherbrooke, QC, Canada, in 1986 and the M.A.S. and Ph.D. degrees from the University of Ottawa, ON, Canada, in 1990 and 1996, all in electrical engineering. His Master and Doctorate degree theses have been conducted in collaboration with the Mobile Satellite Group of the Communications Research Centre (CRC), Ottawa, Canada. In 1995, he joined CRC to work on signal processing applications for communications and for spectrum monitoring. His main research interests include modulation and coding, detection and estimation in the spectrum monitoring context, and real-time signal processing.