FORWARD COMPUTATION OF BACKWARD PATH METRICS FOR MAP DECODER

Yufei Wu, William J. Ebel, and Brian D. Woerner

Mobile and Portable Radio Research Group
432 NEB, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0350, USA
Phone: 001-540-231-2920   Fax: 001-540-231-2968
e-mail: `yufei@vt.edu`

Abstract – In the implementation of the MAP algorithm for turbo decoding, the memory management is an important issue. When the sliding window is not used, either the forward or the backward path metrics need to be stored for the whole frame. In this paper[1], an alternative method is proposed to reduce the storage of the path metrics. Most path metrics are calculated as needed with only a small portion of those metrics drawn from memory. A simulation study is performed to justify the choice of the block size for this method. This technique can reduce the memory requirement by at least 50% without compromising the performance while incurring a small penalty in computation complexity.

I. INTRODUCTION

The maximum *a posteriori* (MAP) algorithm is widely used for implementation of turbo decoders. This algorithm requires a computation of the backward path metric $B_k(s)$ and the forward path metric $A_k(s)$. Either one of them can be computed first and stored in memory. In the backward-forward version, the soft output can be produced with increasing time index by performing the backward recursion first to obtain $B_k(s)$ for all k and s. After that, the forward recursion is performed and the k-th log-likelihood ratio (LLR) is computed using $A_{k-1}(s)$ and $B_k(s)$. Here $k \in \{1, ..., N\}$ is the time index, $s \in \{0, ..., 2^m - 1\}$ is the state index, N is the frame size, and m is the memory size of the constituent recursive systematic convolutional (RSC) encoder. This backward-forward version is adopted for analysis in this paper.

Although only the most recent vector $A_{k-1}(s)$ needs to be kept for the calculation of $A_k(s)$, all $B_k(s)$ must be stored for the computation of LLR. This can be a problem when the frame size is large, since $2^m \times N$ values of $B_k(s)$ need to be stored. Sliding window techniques [1] [2] have been developed to save the memory. However, they do not make use of the information in the whole frame. This paper presents a method to reduce storage, which does make use of the information in the whole frame and computes $B_k(s)$ in a forward manner. As a result, the memory requirement for $B_k(s)$ is relaxed. In addition, this method can be combined with the sliding window technique to achieve even greater savings.

The remainder of the paper is organized as follows. In Section II, a brief review of the MAP algorithm is presented for the purpose of defining notations. In Section III, the composition of the butterfly structure in a good RSC trellis is shown. Making use of the properties of this structure, a matrix representation of the backward path metric recursion is given in Section IV. The forward calculation of $B_k(s)$ is obtained via matrix inversion. In Section V, performance examples are shown to illustrate the effect of different block size. Section VI concludes this paper.

II. CONVENTIONAL FORWARD-BACKWARD MAP ALGORITHM

A general review of the backward-forward MAP algorithm is presented below. In the following, u_k stands for the k-th information bit, $c_k = (c_k^{(1)}, c_k^{(2)})$ represents the codeword, x_k is the transmitted signal corresponding to c_k, $y_k = (y_k^{(1)}, y_k^{(2)})$ is the k-th received signal, $Y_i^j = (y_i, ..., y_j)$ is the portion of received sequence from time i to j, $1 \le i < j \le N$. $P_k(u; I)$ and $P_k(c; I)$ are input probability distributions for the k-th information bit and the k-th codeword respectively. $P_k^A(u; O)$ is the *a posteriori* probability (APP) distribution for the k-th information bit. An edge in an encoder trellis and the notations related to it are introduced in Fig. 1. Notice that $s_k^E(e) = s_{k+1}^S(e)$ when the trellis sections are linked together.

According to the BCJR [3] algorithm, the branch metric on edge e is

$$\begin{aligned} M(e) &= P(s_k^E(e), y_k | s_k^S(e)) \\ &= \sum_{x_k} P(s_k^E(e) | s_k^S(e)) \end{aligned}$$

[1]This work has been supported by the MPRG Industrial Affiliates Foundation and the Defense Advanced Research Projects Agency (DARPA) through the GloMo program.

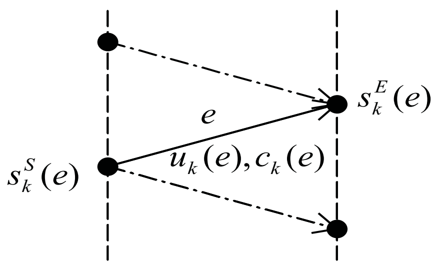

Figure 1: An edge of the trellis.

$$\cdot P(x_k|s_k^S(e), s_k^E(e)) \cdot P(y_k|x_k), \qquad (1)$$

which is a summation over all possible x_k. For a trellis without parallel transition, when $s_k^S(e)$ goes to $s_k^E(e)$ with the output x_k, $P(x_k|s_k^S(e), s_k^E(e)) = 1$. Otherwise it equals to 0. The first term in (1) is given by

$$P(s_k^E(e)|s_k^S(e)) = P(u_k|Y_1^N) = P_k(u(e); I). \qquad (2)$$

If all possible x_k are equally probable, the third term in (1) is

$$\begin{aligned} P(y_k|x_k) &= \frac{P(x_k|y_k)P(y_k)}{P(x_k)} \\ &= h_M P(x_k|y_k) \\ &= h_M P_k(c(e); I), \end{aligned}$$

where $h_M = P(y_k)/P(x_k)$. Thus when the transition exists between $s_k^S(e)$ and $s_k^E(e)$,

$$M(e) = h_M P_k(u(e); I) P_k(c(e); I). \qquad (3)$$

$A_k(\cdot)$ and $B_k(\cdot)$ are computed with forward and backward recursions for all $k \in \{1, ..., N-1\}$:

$$\begin{aligned} A_k(s) &= P(s_k^E(e) = s, Y_1^k) \\ &= \sum_{e:s_k^E(e)=s} A_{k-1}(s_k^S(e))M(e), \qquad (4) \\ B_k(s) &= P(Y_{k+1}^N | s_{k+1}^S(e) = s) \\ &= \sum_{e:s_{k+1}^S(e)=s} B_{k+1}(s_{k+1}^E(e))M(e). \qquad (5) \end{aligned}$$

At time k, the APP is:

$$\begin{aligned} &P_k^A(u; O) = P(u_k = u|Y_1^N) \qquad (6) \\ &= \frac{1}{P(Y_1^N)} \sum_{e:u_k(e)=u} P(s_k^S(e), s_k^E(e), Y_1^N) \\ &= h_c \sum_{e:u_k(e)=u} A_{k-1}(s_k^S(e))M(e)B_k(s_k^E(e)), \end{aligned}$$

where h_c is the constant that can be ignored when the soft output is expressed in the following LLR form:

$$\lambda_k^A(u; O) = \log \frac{P_k^A(u=1; O)}{P_k^A(u=0; O)}.$$

III. EXISTENCE OF BUTTERFLY PAIRS

For a binary system, the branch transitions appear as butterfly pairs when the first and the last shift registers are both connected in the feedback and feed forward polynomials, as shown below.

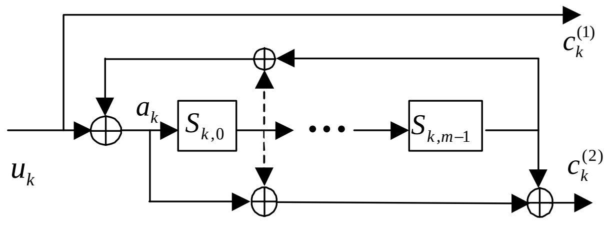

Figure 2: A general RSC encoder.

A rate 1/2 RSC encoder model is drawn in Fig. 2. Denote the state of the encoder as $(S_{k,0}, ..., S_{k,m-1})$, the substate as $(S_{k,0}, ..., S_{k,m-2})$, the information bit as u_k, and the encoder output as $c_k = (c_k^{(0)}, c_k^{(1)})$, where $c_k^{(0)}$ and $c_k^{(1)}$ are the systematic and the parity bits, respectively, k is the time index. The following observations are made, for a given substate $(S_{k,0}, ..., S_{k,m-2})$.

1. *The relationship between the current state and the next state.* If the feedback polynomial has a connection to both u_k and $S_{k,m-1}$, then the opposite value of $u_k \oplus S_{k,m-1}$ will correspond to the opposite value of a_k. Since a_k will be shifted to the first register and become $S_{k+1,0}$ at the next cycle, the next state is determined as a result. Or, with $A \in \{0, 1\}$,

$$\left\{ \begin{array}{l} u_k \oplus S_{k,m-1} = 0 \\ u_k \oplus S_{k,m-1} = 1 \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} a_k = A \\ a_k = \bar{A} \end{array} \right.$$
$$\Longrightarrow \text{next state at time k+1 :} \left\{ \begin{array}{l} (A, S_{k,0}, ..., S_{k,m-2}) \\ (\bar{A}, S_{k,0}, ..., S_{k,m-2}) \end{array} \right.$$

Define the index of state as

$$I = S_{k,0} \times 2^{m-1} + \cdots + S_{k,m-2} \times 2 + S_{k,m-1},$$

and let $\quad M = S_{k,0} \times 2^{m-2} + \cdots + S_{k,m-2}$, then the state pair $2M$ (when $S_{k,m-1} = 0$) and $2M+1$ (when $S_{k,m-1} = 1$) always lead to the next state pair M (when $a_k = 0$) and $2^{m-1} + M$ (when $a_k = 1$).

2. *The codewords in a butterfly pair.* The value of $a_k \oplus S_{k,m-1}$ is related to the value of $c_k^{(1)}$ via the equation

$$\left\{ \begin{array}{l} a_k + S_{k,m-1} = 0 \\ a_k + S_{k,m-1} = 1 \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} c_k^{(1)} = C \\ c_k^{(1)} = \bar{C} \end{array} \right. ,$$

where $C \in \{0, 1\}$. Since $S_{k,m-1}$ is connected to a_k by the backward polynomial, and both of them are connected to $c_k^{(1)}$ by the forward polynomial, the influence of $S_{k,m-1}$ on $c_k^{(1)}$ is canceled. The substate $(S_{k,0}, ..., S_{k,m-2})$ will uniquely determine whether $c_k^{(1)} = u_k$ or $c_k^{(1)} = \bar{u}_k$. Considering $c_k^{(0)} \equiv u_k$, there are two possible codewords for a given substate: $(c_k^{(1)}, c_k^{(2)}) \in \{(u_k, u_k), (u_k, \bar{u}_k)\}$. For each butterfly pair of states, one choice is made since the current states share the same substate. In other words, even though there are four branches in a butterfly pair, there can be only two values of codeword for a butterfly pair:(00),(11) for (u_k, u_k) case, or (01),(10) for $(u_k, \bar{u}_k)$ case.

To summarize, a RSC code generator which has connections to u_k and $S_{k,m-1}$ for both forward and backward polynomials has a trellis which can be grouped into 2^{m-1} butterfly pairs. Each butterfly is determined by a unique substate. Good RSC encoders for turbo codes always satisfy this condition. When the rate is 1/2, half of the pairs have codewords (00) and (11), half of them have codewords (01) and (10). Each pair assumes the form in Fig. 3.

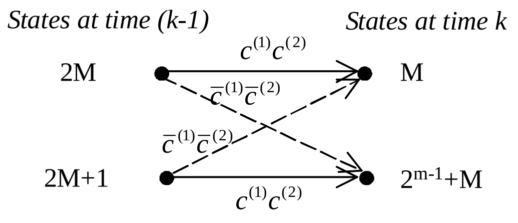

Figure 3: The model of butterfly M.

IV. FORWARD RECURSIVE COMPUTATION OF $B_K(S)$

Using the butterfly structure shown in Fig 3, the MAP algorithm can be grouped into 2^{m-1} sets of 2 × 2 matrix operations, one for each butterfly pair. Here a rate 1/2 RSC encoder with code generator $g(D) = [1, 1+D^2/1+D+D^2]$ is used as an example. A trellis section for this code generator at time k is shown at Fig. 4. The two butterfly pairs match the form given in Fig. 3.

Corresponding to four different codewords, there are four different branch metric values. Omit the constant h_M in (3), and represent the trimmed branch metric at time k by $G_k(c(e))$, where $c(e)$ is the codeword on edge e. There are two branch metrics for butterfly pair 1:

$$\begin{aligned} G_k(00) &= P_k(u_k = 0; I)P_k(c_k = 00; I), \\ G_k(11) &= P_k(u_k = 1; I)P_k(c_k = 11; I). \end{aligned}$$

And two branch metrics for butterfly pair 2:

$$\begin{aligned} G_k(01) &= P_k(u_k = 0; I)P_k(c_k = 01; I), \\ G_k(10) &= P_k(u_k = 1; I)P_k(c_k = 10; I). \end{aligned}$$

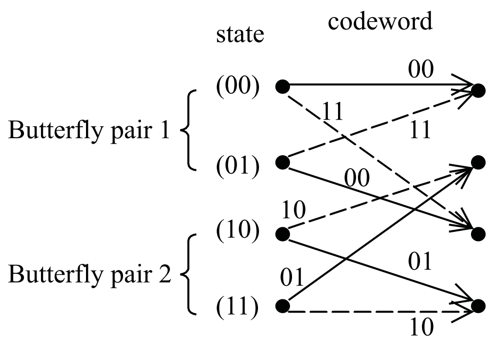

Figure 4: A trellis section with $g(D) = [1, 1+D^2/1+D+D^2]$.

The backward path metrics $B_k(s)$ are updated by the following equations. For butterfly pair 1:

$$\begin{aligned} B_k(0) &= G_k(00)B_{k+1}(0) + G_k(11)B_{k+1}(2), \\ B_k(1) &= G_k(11)B_{k+1}(0) + G_k(00)B_{k+1}(2), \end{aligned}$$

which may be expressed in matrix form as

$$\begin{aligned} \begin{bmatrix} B_k(0) \\ B_k(1) \end{bmatrix} &= \begin{bmatrix} G_k(00) & G_k(11) \\ G_k(11) & G_k(00) \end{bmatrix} \begin{bmatrix} B_{k+1}(0) \\ B_{k+1}(2) \end{bmatrix} \\ &= G_{k1}^B \begin{bmatrix} B_{k+1}(0) \\ B_{k+1}(2) \end{bmatrix}. \end{aligned} \quad (7)$$

And for butterfly pair 2:

$$\begin{aligned} B_k(2) &= G_k(10)B_{k+1}(1) + G_k(01)B_{k+1}(3) \\ B_k(3) &= G_k(01)B_{k+1}(1) + G_k(10)B_{k+1}(3) \end{aligned}$$

which may be expressed in matrix form as

$$\begin{aligned} \begin{bmatrix} B_k(2) \\ B_k(3) \end{bmatrix} &= \begin{bmatrix} G_k(10) & G_k(01) \\ G_k(01) & G_k(10) \end{bmatrix} \begin{bmatrix} B_{k+1}(1) \\ B_{k+1}(3) \end{bmatrix} \\ &= G_{k2}^B \begin{bmatrix} B_{k+1}(1) \\ B_{k+1}(3) \end{bmatrix}. \end{aligned} \quad (8)$$

If $(G_{k1}^B)^{-1}$ and $(G_{k2}^B)^{-1}$ exist, $B_{k+1}(\cdot)$ can be obtained from $B_k(\cdot)$ based on (7) and (8). For butterfly pair 1:

$$\begin{aligned} &\begin{bmatrix} B_{k+1}(0) \\ B_{k+1}(2) \end{bmatrix} = (G_{k1}^B)^{-1} \begin{bmatrix} B_k(0) \\ B_k(1) \end{bmatrix} \\ &= h_G \times (G_k^2(10) - G_k^2(01)) \times \\ &\begin{bmatrix} G_k(00) & -G_k(11) \\ -G_k(11) & G_k(00) \end{bmatrix} \begin{bmatrix} B_k(0) \\ B_k(1) \end{bmatrix} \end{aligned} \quad (9)$$

For butterfly pair 2:

$$\begin{aligned} &\begin{bmatrix} B_{k+1}(1) \\ B_{k+1}(3) \end{bmatrix} = (G_{k2}^B)^{-1} \begin{bmatrix} B_k(2) \\ B_k(3) \end{bmatrix} \\ &= h_G \times (G_k^2(00) - G_k^2(11)) \times \\ &\begin{bmatrix} G_k(10) & -G_k(01) \\ -G_k(01) & G_k(10) \end{bmatrix} \begin{bmatrix} B_k(2) \\ B_k(3) \end{bmatrix} \end{aligned} \quad (10)$$

In the above, $h_G = \frac{1}{(G_k^2(10)-G_k^2(01))(G_k^2(00)-G_k^2(11))}$. If LLR is computed as the output, h_G will be canceled. Thus it can be omitted.

Equations (9) and (10) show that $B_{k+1}(s)$ can actually be determined from $B_k(s)$. This suggests that the backward recursion can be performed, without storing any $B_k(s)$, to obtain $B_1(s)$. Then (9) and (10) can be employed to compute $B_k(s)$ simultaneously with the $A_{k-1}(s)$ to obtain the soft output.

To assure the computational stability of the matrix inverse operation in (9) and (10), we propose dividing the frame into blocks of $N_b(\geq 2)$ bits, storing only the $B_k(s)$ at the beginning of each block during backward recursion, and using (9) and (10) to compute the $B_k(s)$ within each block during the forward recursion. The choice of N_b will be discussed in Section V. In summary, the modified decoding procedure is described as follows:

1. Perform the conventional backward recursion using (7) and (8), but store only $B_j(s)$, where $j = 1$ or iN_b, $0 < i < N_{blk}$, and $N_{blk} = \lceil N/N_b \rceil$. Set $k = 1$.

2. Perform the forward recursion of the conventional MAP algorithm to obtain $A_{k-1}(s)$. Use the stored $B_k(s)$ for $k = 1$ and $iN_b, 0 < i < N_{blk}$. Otherwise, calculate $B_k(s)$ from the previous vector $B_{k-1}(s)$ using (9) and (10), similar to the computation of $A_k(s)$.

3. Use $B_k(s)$ in conjunction with $A_{k-1}(s)$ to evaluate the k-th soft output. Go back to Step 2 if $k < N$.

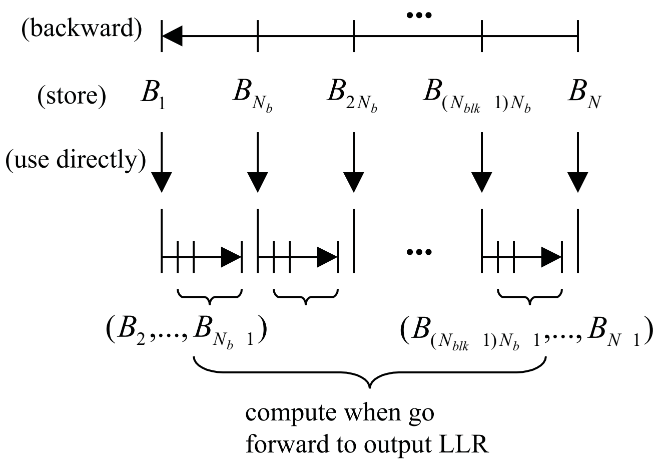

Figure 5: Procedure to compute $B_k(s)$.

The procedure to compute $B_k(s)$ is illustrated in Fig. 5. With this method, a factor of $(N_b - 1)/N_b$ storage space for $B_k(s)$ will be saved with the price of computation complexity. The computation of $B_k(s)$ is increased to $(1+(N_b-1)/N_b)$ times that of the conventional MAP algorithm. Notice that when $N_b = 1$, this method degenerates to be the conventional MAP algorithm.

V. CHOICE OF BLOCK SIZE

As mentioned above, numerical stability problems place constraints upon the choice of the block size N_b. Using the previous example of $g(D) = [1, 1 + D^2/1 + D + D^2]$, if (7) and (8) are used iteratively to compute $B_i(s)$ from $B_{i+l}(s)$ for $1 \leq l \leq N - i$, the following expression linking the path metrics at $k = i$ and $k = i + l$ can be obtained:

$$\begin{bmatrix} B_i(0) \\ B_i(1) \\ B_i(2) \\ B_i(3) \end{bmatrix} = G^B_{i,i+l} \begin{bmatrix} B_{i+l}(0) \\ B_{i+l}(1) \\ B_{i+l}(2) \\ B_{i+l}(3) \end{bmatrix},$$

where $G^B_{i,i+l}$ is a 4×4 matrix. Therefore, we compute $B_{i+l}(s)$ from $B_i(s)$ using the following:

$$\begin{bmatrix} B_{i+l}(0) \\ B_{i+l}(1) \\ B_{i+l}(2) \\ B_{i+l}(3) \end{bmatrix} = (G^B_{i,i+l})^{-1} \begin{bmatrix} B_i(0) \\ B_i(1) \\ B_i(2) \\ B_i(3) \end{bmatrix}.$$

The element $G^B_{i,i+l}(p,q), 0 \leq p,q \leq 3$, is proportional to the probability of going from state p at time i to state q at time $i + l$, or $G^B_{i,i+l}(p,q) = P(s_{i+l} = q, Y^{i+l}_{i+1} | s_i = p)$. In consequence, if l is larger than the time the trellis needs to converge, the matrix $G^B_{i,i+l}$ will have an entry much larger than the others, which corresponds to the most likely path in the trellis section between $k = i$ and $k = i + l$. Mathematically, $G^B_{i,i+l}$ becomes nearly singular, and $(G^B_{i,i+l})^{-1}$ becomes ill-conditioned. From another perspective, as the backward recursion is performed, the information from stage to stage is aggregated together. When l increases, it becomes increasingly difficult to retrieve the detailed information for each transition.

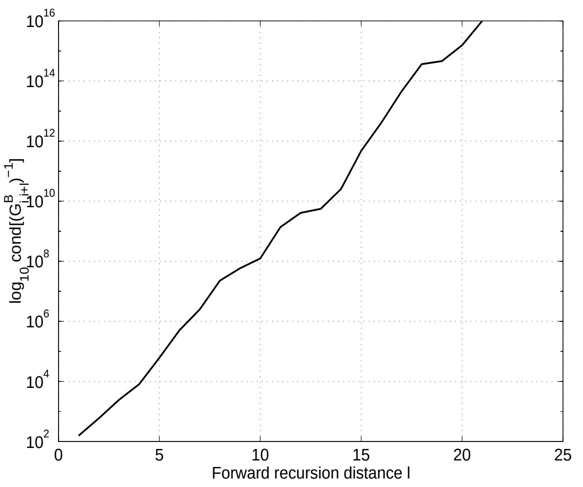

Figure 6: Logarithm of the condition number.

A typical example of $g(D) = [1, 1+D^2/1+D+D^2]$ in an AWGN channel at $E_b/N_0 = 1$ dB is used throughout the following simulations. A plot of the logarithm of the condition number of $(G^B_{i,i+l})^{-1}$ is shown in Fig. 6 for one MAP operation. The depth l ranges from 1 to $7K$, where $K = m+1 = 3$ is the constraint length. As seen in the plot, the condition number of the forward matrix $(G^B_{i,i+l})^{-1}$ increases exponentially with the distance l.

Let $\lambda^A_k(u;O)$ be the k-th soft output obtained with the conventional method, $\lambda^{A,N_b}_k(u;O)$ be the one obtained with the new method and block size N_b. The relative error is defined to be

$$\Delta^{N_b}_r = \frac{|\lambda^A_k(u;O) - \lambda^{A,N_b}_k(u;O)|}{E(|\lambda^A_k(u;O)|)},$$

where $E(x)$ is the mean value of x. In Fig. 7, the maximum $\Delta^{N_b}_r$ for a frame of size 600 is shown with N_b ranging from $3K$ to $10K$. It shows that the relative error is negligible when $N_b \leq 5K$. When $N_b \geq 6K$, the computation error accumulates to an unacceptable value.

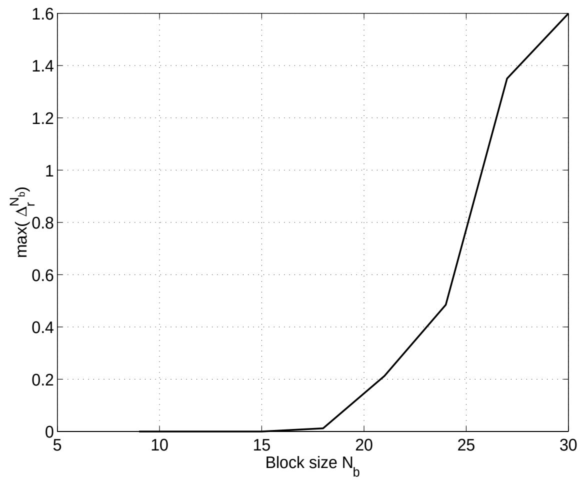

Figure 7: Maximum relative error of LLR.

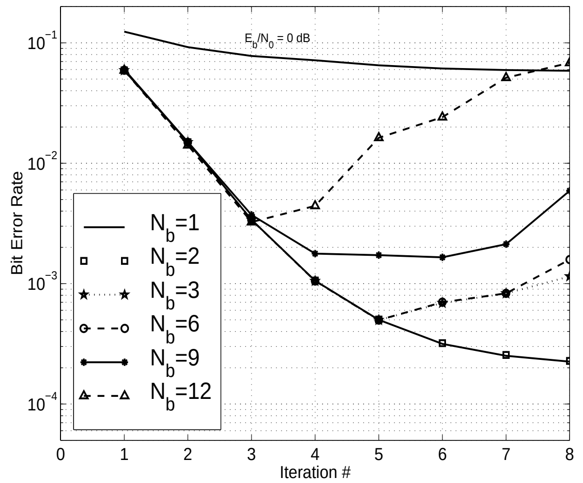

Figure 8: N_b choice for multiple iterations ($E_b/N_0 =$ 1 dB).

While Fig. 7 indicates that the forward recursive computation of $B_k(s)$ have negligible effect on numerical stability for a single iteration, small errors may accumulate over multiple iterations, severely degrading performances. This effect may limit the size of N_b used in practice. In Fig. 8, curves are shown for BER vs. iteration number. At $E_b/N_0 = 0$ dB, $N_b \in \{1, ..., 12\}$ gives the same BER. At $E_b/N_0 = 1$ dB, BER decreases steadily for the first few iterations and then increases when $N_b \in \{3, ..., 12\}$. As explained above, this is due to the convergence speed of the trellis. As the number of iterations increases, the decoder becomes more confident about the estimation. At medium to high E_b/N_0, N_b needs to be set smaller when the MAP decoder converges rapidly. For a decoder working with medium to high E_b/N_0 and/or a large number of iterations, it may become necessary to set $N_b = 2$, which saves 50% of the storage for $B_k(s)$.

VI. CONCLUSION

In this paper, the existence of butterfly pairs for the RSC trellis is shown. Based on this structure, a new method is shown to compute the backward path metrics $B_k(s)$ in a forward manner for the MAP algorithm. This method promises to save at least 50% memory size of the backward path metrics $B_k(s)$ without performance degradation. This is advantageous for the implementation of MAP decoder for the turbo codes, especially when the frame size is large. This offers the possibility to store $B_k(s)$ on chip. In comparison to off-chip storage, it can speed up the processing by reducing the clock cycles required to fetch data. In some cases this advantage may be sufficient to compensate for the increased computation required by the technique.

REFERENCES

[1] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-output decoding algorithms in iterative decoding of turbo codes," *JPL TDA Progress Report 42-124*, Feb. 15, 1996.

[2] A. J. Viterbi, "An intuitive justification and a simplified implemetation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[3] 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. IT-20, pp. 284–287, Mar. 1974.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," *JPL TDA Progress Report 42-127*, Nov. 15, 1996.

[5] D. Taipale, *Implementing Viterbi Decoders Using the VSL Instruction on DSP Families DSP56300 and DSP56600.* Semiconductor Products Sector, Motorola, May 1998.