A MAP equivalent SOVA for Non-binary Turbo Codes

Jun Tan and Gordon L. Stüber
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332-0250, USA

***Abstract*—An optimal MAP-equivalent SOVA decoding algorithm and its simplified suboptimal algorithm for non-binary codes are proposed. The implementation of its suboptimal algorithm is simpler, while its performance is very close to the optimal Log-MAP algorithm. The proposed SOVA can be used as a decoder for Turbo trellis coded modulation (TTCM). Simulation results for both the proposed SOVA and the Log-MAP algorithms for TTCM and binary Turbo codes are presented. It is concluded that the proposed SOVA performs very close to the Log-MAP algorithm for both the TTCM and binary Turbo codes, and its performance is better than the conventional SOVA.**

I. Introduction

Recent research in the detection and decoding of digital signals in the presence of interference and noise leads to the soft-output, instead of hard-output, detection and decoding algorithms. The soft-output provides possibilities for concatenated/iterative detection or decoding. Thus it could improve the detection and decoding performance as a whole. One of the soft-output algorithms, the symbol-by-symbol maximum *a posteriori* (MAP) algorithm [2] gives optimum performance with minimum sysmbol error probability. It was proposed about three decades ago, but only recently its applications in concatenated codes (such as Turbo codes[5]) and channel equalization [7] draw a lot of attention. The other soft-output algorithm, the Soft-Output Viterbi Algorithm (SOVA) [3], [4], is a modified Viterbi Algorithm (VA) with soft output values. It was also applied for Turbo codes decoding as an alternative decoding algorithm [6]. The SOVA is actually a sub-optimal MAP algorithm, equivalent to the Max-Log-MAP algorithm [14].

Because of the complexity of MAP algorithm, the implementation of MAP is always proposed in logarithm domain, where the MAP algorithm becomes Log-MAP algorithm. Further simplification to Log-MAP have been proposed, such as a look-up table approximation [8], and Max-Log-MAP [8]. There are two recursion processes in the MAP algorithm, the forward and the backward recursion. In an elegant interpretation of Max-Log-MAP algorithm [13], the forward and backward recursions are simplified as forward and backward VA. Together with the two VA recursions gives the soft output. There is another class of MAP algorithms, called "continuous" algorithms [10], which estimates the soft output after a given delay with a sliding window. In the sliding window, both forward and backward operations are performed for the soft output values. In a forward-recursion-only MAP algorithm proposed in [7], the soft output is estimated after a certain delay. Since no backward recursion operations are needed, such MAP algorithm is suitable for the detecting algorithm of equalizers.

This research was supported by the National Science Foundation under grant CCR-9903297.

The MAP and Log-MAP algorithms are optimal in the sense of maximum *a posteriori* sequence probability. The simplified versions of MAP or Log-MAP, such as Max-Log-MAP, SOVA, are suboptimal. The optimality are traded for simplification of implementation. It is reported that SOVA is roughly half as complex as the Log-MAP [8] with some performance degradation. As for non-binary codes, the complexity of MAP algorithm becomes overwhelming and the simplified suboptimal algorithms are badly needed, with desirable small performance degradation. In this paper, a different implementation of MAP algorithm is proposed, together with a suboptimal algorithm. It is shown the performance of the suboptimal algorithm for non-binary Turbo codes is near to the performance of optimal MAP.

This paper is organized as follows. Section II gives an introduction on the Turbo TCM, which would be applied for non-binary decoding simulation in section V. In section III, we review the Viterbi Algorithm and MAP algorithm. The proposed MAP equivalent algorithm, together with a simplified algorithm, is derived in section IV. Section V gives some simulation results. Section VI concludes the presentation.

II. TTCM Encoder and Decoder

The main ingredients of a Turbo coding structure are the use of recursive systematic convolutional (RSC) codes and pseudo-random interleavers. It is well-known that trellis codes, which can be represented by a recursive convolutional encoder followed by a signal mapper, result in a coding gain without bandwidth expansion. Therefore, the motivation for the Turbo TCM encoder structure comes from the fact that by replacing the component codes in a binary Turbo encoder with trellis codes, we should be able to achieve good coding gains without bandwidth expansion.

An example Turbo TCM encoder[12] is shown in Fig. 1. The binary source data is first split into m parallel streams

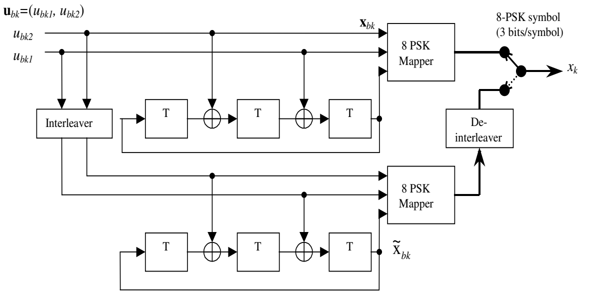

Fig. 1. Turbo TCM Encoder, from [12]

and groups of m data bits form an input symbol. The input symbol at epoch k can be represented by $\mathbf{u}_{bk} = (u_{bk0}, u_{bk1}, ..., u_{bkm-1})$, where each u_{bki} is a source bit. In the sequel, the binary vector $\mathbf{u}_{bk}$ is represented as a 2^m-ary symbol u_k, i. e. , $u_k = a$ for $a = 0, 1, ..., 2^m - 1$. The input vector $\mathbf{u}_{bk}$ is encoded by a rate $m/m+1$ RSC encoder. The $m+1$ binary output of the convolutional encoder at epoch k, is $\mathbf{x}_{bk} = (x_{bk0}, x_{bk1}, ..., x_{bkm-1}, x_{bkm}) = (\mathbf{u}_{bk}, x_{bkm})$, which contains parity bit x_{bkm} and the systematic part $\mathbf{u}_{bk}$. The encoder output $\mathbf{x}_{bk}$ is used to select a signal point x_k from a 2^{m+1}-PSK signal constellation. The interleaver in the encoder is a pairwise random interleaver with one constraint: even position to even position, odd position to odd position. The interleaved sequence is then encoded by using another rate $m/(m+1)$ encoder. The encoder output symbols are punctured and transmitted.

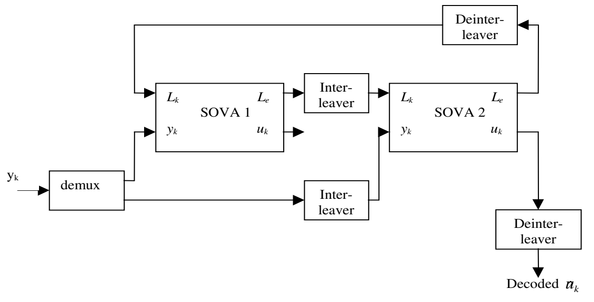

Fig. 2. Iterative Decoder with SOVA

Fig. 2 shows a decoder for Turbo codes that uses SOVA component decoders. The input sequence to the decoder is denoted by $\{y_k\}$. The output $\hat{u}_k$ of the iterative decoder represents the decoded symbol. The *a priori* information to each decoder is L_k, while the extrinsic information is L_e. L_e is also called the soft-output value. The first component decoder, SOVA1, decodes for the first component encoder, while SOVA2 for the second component encoder.

III. Review on VA and MAP algorithm

Review of the Viterbi Algorithm: The Viterbi Algorithm (VA) [1] in its most general form is a maximum *a posteriori* probability sequence estimator, which finds the maximum *a posteriori* probability path over the trellis diagram given the received sequence $\mathbf{y} = \{y_k\}_{k=1}^N$. Here the symbol block length or symbol frame length is N, while the binary block length is mN. A path in the trellis can be represented by a state sequence $\mathbf{s} = \{s_1, s_2, ..., s_N\}$, which indicates the trellis path starting at state s_1, passing through every state s_k at epoch k, and terminating at state s_N. The VA finds the trellis path or state sequence $\mathbf{s}$ so that the *a posteriori* probability $P(\mathbf{s}|\mathbf{y})$ is maximized. Assume the state sequence $\mathbf{s}$ is a Markov sequence. Since the received sequence $\mathbf{y} = \{y_k\}_{k=1}^N$ does not dependent on the selection of trellis path $\mathbf{s}$, and $P(\mathbf{s}|\mathbf{y}) = p(\mathbf{s}, \mathbf{y})/p(\mathbf{y})$, at epoch k, we can equivalently maximize

$$p(\mathbf{s}_k, \mathbf{y}_k) = p(\mathbf{s}_{k-1}, \mathbf{y}_{k-1}) P(u_k) p(y_k|s', s), \qquad (1)$$

where $\mathbf{s}_k = \{s_1, s_2, ..., s_k\}$, $\mathbf{y}_k = \{y_1, y_2, ..., y_k\}$, $s' = s_{k-1}$, $s = s_k$, and u_k is the source symbol for the state transition $s' \to s$ of trellis path $\mathbf{s}_k$. The path metric $M_k(\mathbf{s}_k)$ associated with the trellis path $\mathbf{s}_k$ is defined as

$$M_k(\mathbf{s}_k) = \log(p(\mathbf{s}_k, \mathbf{y}_k)). \qquad (2)$$

Obviously,

$$p(\mathbf{s}_k, \mathbf{y}_k) = \exp\big(M_k(\mathbf{s}_k)\big). \qquad (3)$$

Substituting (1) into (2) gives

$$M_k(\mathbf{s}_k) = M_{k-1}(\mathbf{s}_{k-1}) + \log P(u_k) + \log(p(y_k|s', s)). \quad (4)$$

Using the notation $\lambda = log(P)$, (4) becomes

$$M_k(\mathbf{s}_k) = M_{k-1}(\mathbf{s}_{k-1}) + \lambda(u_k) + \lambda(y_k|s', s). \qquad (5)$$

In (5), $\lambda(u_k)$ is the *a priori* information of the source symbol u_k, and $\lambda(y_k|s', s)$ is the branch metric for the state transition $s' \to s$ given the received signal y_k. At epoch k, for each state s, the path metrics for all possible paths terminating at state s are calculated. Only the maximal path metric is saved and the corresponding path is the survivor path.

Review of MAP algorithm: Although it is a MAP estimator in its most general form [1], the VA is generally considered as a maximal likelihood (ML) estimator. The only difference between MAP and ML is the *a priori* information. When the probability of source information is equally distributed, or it is justified assumed equally, MAP and ML are equivalent. VA performs as a ML estimator because the *a priori* source information is almost not available. For the applications in concatenated and iterative detecting/decoding, the *a priori* information are needed for further processing in order to gain better performance. The MAP estimator becomes essential.

The MAP algorithm, following its name, finds the maximum *a posteriori* sequence probability $P(\hat{\mathbf{u}}|\mathbf{y})$. The *a posteriori* probability at epoch k is generally defined as[11], [2],

$$P(\hat{u}_k = a|\mathbf{y}) = \sum_{\substack{(s',s)\\ u_k=a}} p(s', s, \mathbf{y}), \tag{6}$$

where (s', s) is the state transition at epoch k, and $a = 0, 1, ..., 2^m - 1$. (6) indicates the estimation of the *a posteriori* probability happens at epoch k. However, (6) is not the only definition of the *a posteriori* probability. There are some other MAP algorithms perfer to estimate the *a posteriori* probability after a time delay [10], [7]. In the appendix of [7], two types of MAP algorithms are summarized.

One elegant derivation of the *a posteriori* probability was presented by splitting the joint probability[11], as

$$\begin{aligned} p(s', s, \mathbf{y}) &= p(s', s, \mathbf{y}_{j<k}, \mathbf{y}_k, \mathbf{y}_{j>k}) \\ &= p(s', \mathbf{y}_{j<k}) \cdot p(\mathbf{y}_k, s|s') \cdot p(\mathbf{y}_{j>k}|s) \\ &= \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s). \end{aligned} \tag{7}$$

The splitting of the joint probability leads to three items: a branch metric, a forward recursion, and a backward recursion. The forward recursion yields,

$$\alpha_k(s) = \sum_{s'} \gamma_k(s', s) \cdot \alpha_{k-1}(s'), \tag{8}$$

and the backward recursion yields,

$$\beta_{k-1}(s') = \sum_{s} \gamma_k(s', s) \cdot \beta_k(s). \tag{9}$$

The $\gamma_k(s', s)$ is the branch metric for state transition $s' \to s$ at epoch k.

IV. MAP equivalent SOVA for non-binary codes

As we discussed before, the *a posteriori* probability of MAP algorithm has several definitions. Here we follow the definition in (6). Different from the derivation of $p(s', s, \mathbf{y})$ in (7), there is another way to derive the joint probability $p(s', s, \mathbf{y})$, as

$$\begin{aligned} p(s', s, \mathbf{y}) &= P(s', s|\mathbf{y})p(\mathbf{y}) \\ &= P(s'|s, \mathbf{y})P(s|\mathbf{y})p(\mathbf{y}), \end{aligned} \tag{10}$$

where $p(\mathbf{y})$ is similiar for all state transitions. The probability of s' given s and the received $\mathbf{y}$ is

$$P(s'|s, \mathbf{y}) = \frac{p(\mathbf{s}_k^{(s'\to s)}, \mathbf{y}_{j<k})}{\sum_{s''} p(\mathbf{s}_k^{(s''\to s)}, \mathbf{y}_{j<k})}, \tag{11}$$

where $\mathbf{s}_k^{(s'\to s)}$ is the trellis path containing branch $(s' \to s)$, and $\mathbf{s}_k^{(s''\to s)}$ is the trellis path containing branch $(s'' \to s)$ at epoch k. For each state s at k, the number of trellis branches terminated at s is equal to M for M-ary codes. There are M possible states s'' for state transitions $s'' \to s$. The ratio of probability of a trellis path containing branch $(s' \to s)$ to the sum of probabilities of all possible trellis paths terminated at s is the probability of state s' given s and $\mathbf{y}$, i.e., $P(s'|s, \mathbf{y})$. From (3),

$$P(s'|s, \mathbf{y}) = \frac{\exp\left(M_k(\mathbf{s}_k^{(s'\to s)})\right)}{\sum_{s''} \exp\left(M_k(\mathbf{s}_k^{(s''\to s)})\right)}, \tag{12}$$

where the received symbol sequence $\mathbf{y}_{j>k}$ is independent of the state transition at epoch k. The conditional probability $P(s|\mathbf{y})$ can be yielded through backward recursion, as

$$P(s'|\mathbf{y}) = \sum_{s} P(s'|s, \mathbf{y})P(s|\mathbf{y}). \tag{13}$$

The initial value of the backward recursion is $P(s|\mathbf{y}) = 1$ for terminal state s at epoch $k = N$, and $P(s|\mathbf{y}) = 0$ otherwise.

From the soft output definition in (6), the soft output should have M possible values for M-ary source symbols. The soft output value of the proposed algorithm in probability form is obtained as,

$$P(\hat{u}_k = a|\mathbf{y}) = \sum_{\substack{(s'\to s)\\ u_k=a}} P(s|\mathbf{y}) \frac{\exp\left(M_k(\mathbf{s}_k^{(s'\to s)})\right)}{\sum_{s''} \exp\left(M_k(\mathbf{s}_k^{(s''\to s)})\right)}. \tag{14}$$

Using the joint probability in (10) with the *a posteriori* probability definition in (6), we derived the implementation of MAP algorithm. In our discussion, the path metrics are computed with VA. (6) is the *a posteriori* probability, or the soft output definition of VA. So we still can call the proposed MAP implementation as a SOVA. However, the proposed SOVA is different from the SOVA in literature [4], [6], [15], [14]. The proposed SOVA is an optimal MAP algorithm, while the conventional SOVA is a Max-Log-MAP equivalent, which is sub-optimal.

Simplification of the algorithm: Similar to the MAP algorithm, the proposed SOVA soft output in (14) can be simplified by passing to the log-domain of probability. This leads to the problem of evaluating the logarithm of a sum of exponentials $\log\{\sum_i e^{A_i}\}$. Define $A_{\max} = \max_i\{A_i\}$, then we have,

$$\log(\sum_i e^{A_i}) = A_{\max} + \log(1 + \sum_{A_i \neq A_{\max}} e^{A_i - A_{\max}}). \tag{15}$$

There are two possible approximations to equation (15). The first approach computes the second term in the right-hand side of (15) using look-up tables[8] . Another possible approximation, which is also used in Max-Log-MAP algorithm, just neglects the second term. The discussion below follows the latter approach.

Using the log-domain denotation $\lambda = \log P$, and applying the approximation $\log\{\sum_i e^{A_i}\} \approx A_{\max}$ to (6), yields

$$\lambda(\hat{u}_k = a|\mathbf{y}) \approx \max_{\substack{(s' \to s) \\ u_k = a}} \{\lambda(s', s|\mathbf{y})\}. \tag{16}$$

Using (13) and (16), the soft output of the proposed SOVA in log-domain form is,

$$\lambda(\hat{u}_k = a|\mathbf{y}) \approx \max_{\substack{(s' \to s) \\ u_k = a}} \{\lambda(s'|s, \mathbf{y}_N) + \lambda(s|\mathbf{y})\}. \tag{17}$$

The approximations to $\lambda(s'|s,\mathbf{y})$ and $\lambda(s|\mathbf{y})$ and can be obtained through equations (12) and (13), respectively, as

$$\lambda(s'|s,\mathbf{y}) \approx M_k(\mathbf{s}_k^{(s' \to s)}) - \max_{s''}\{M_k(\mathbf{s}_k^{(s'' \to s)})\}. \tag{18}$$

$$\lambda(s'|\mathbf{y}) \approx \max_{s}\{\lambda(s'|s,\mathbf{y}) + \lambda(s|\mathbf{y})\}, \tag{19}$$

The path metric $M_k(\mathbf{s}_k)$ can be obtained by (5). For M-ary symbols, each symbol has M soft output values, corresponding to $a = 0, 1, ..., M-1$. Here, the *a priori* information is $\lambda(u_k)$. The soft output is $\lambda(\hat{u}_k) = \lambda(\hat{u}_k = a|\mathbf{y})$. The extrinsic information, which is the information passed to the next iterative decoder, is $\lambda_e(u_k) = \lambda(\hat{u}_k) - \lambda(u_k)$.

Decoding for binary codes: Although (17) is derived for non-binary codes, it can still apply to the binary codes as a special case. In [11] and [9], Hagenauer et. al. used "Log-likelihood ratio" $L(u_k) = \log\frac{P(u_k=+1)}{P(u_k=-1)}$ to represent the soft output for both binary MAP and SOVA codes. The soft output of the conventional SOVA in [11] is approximated as

$$L(\hat{u}_k) \approx \hat{u}_k \cdot \min_{l=k,\cdots,\delta} \Delta_k^l, \tag{20}$$

where $\Delta_k^l = M_l\left(\mathbf{s}_l^{(survivor)(\hat{u}_k)}\right) - M_l\left(\mathbf{s}_l^{(non-survivor)(-\hat{u}_k)}\right)$ is the metric difference at epoch l, and $\mathbf{s}_l^{(survivor)(\hat{u}_k)}$ is the survivor trellis path terminated at s_l wich contains the branch $s' \to s$ with its corresponding $u_k = \hat{u}_k$; and $\mathbf{s}_l^{(non-survivor)(-\hat{u}_k)}$ is the non-survivor path terminated at s_l wich contains the branch $s' \to s$ with its corresponding $u_k = -\hat{u}_k$.

To compare the proposed SOVA with the conventional SOVA, the Log-likelihood ratio, instead of Log-probability, is used to represent the soft output value for binary codes. Either the Log-probability or the Log-likelihood ratio can represent the soft output for binary codes. From equation (17), the new soft output in Log-likelihood ratio becomes

$$\begin{aligned} L(\hat{u}_k) &= \lambda(\hat{u}_k = +1|\mathbf{y}) - \lambda(\hat{u}_k = -1|\mathbf{y}) \\ &\approx \max_{\substack{s' \to s \\ u_k = +1}} \{\lambda(s'|s,\mathbf{y}) + \lambda(s|\mathbf{y})\} \\ &\quad - \max_{\substack{s' \to s \\ u_k = -1}} \{\lambda(s'|s,\mathbf{y}) + \lambda(s|\mathbf{y})\}, \end{aligned} \tag{21}$$

where, from (19) and (18),

$$\lambda(s'|\mathbf{y}) \approx \max_{s}\{\lambda(s'|s,\mathbf{y}) + \lambda(s|\mathbf{y})\}, \tag{22}$$

and

$$\lambda(s'|s,\mathbf{y}) \approx \begin{cases} 0 & s' \to s \text{ is the survivor branch} \\ -\Delta_k^{(s)} & s' \to s \text{ is not the survivor branch} \end{cases} \tag{23}$$

where $\Delta_k^{(s)} = M_k(\mathbf{s}_k^{(\text{survivor})}) - M_k(\mathbf{s}_k^{(\text{nonsurvivor})})$ is the path metric difference at state s at epoch k. Since the approximation is involved, the simplified algorithm in (21) is sub-optimal. From (21), we can see the soft output of the new SOVA is quite different from that of binary SOVA of [11] and [9].

Comparison with SOVA and Max-Log-MAP: Unlike the conventional SOVA, the proposed SOVA does not need the storage of reliability measures for all states. The soft output values are calculated by forward and backward process. This is different from the updating process for the reliability measures in the conventional SOVA. For binary codes, the simplified proposed SOVA needs the storage of metric difference $\Delta_k^{(s)}$ at every state. This is similiar to the Max-Log-MAP, which requires the entire state metric history be stored. However, in the backward recursion, the proposed SOVA updates the backward value $\lambda(s'|\mathbf{y})$ with the stored metric difference $\Delta_k^{(s)}$, while the Max-Log-MAP needs a full backward "VA" operation. The simplified proposed SOVA only needs one "VA" operation for the calcuation of metrics. It is less complex than the Max-Log-MAP algorithm.

V. Simulation Results

The simulation for non-binary Turbo codes is based on Robertson's Turbo TCM encoder [12] shown in Fig. 1. The component encoders are recursive rate-2/3 convolutional encoders with $2^3 = 8$ states. Each source symbol contains 2bits. The 2-bit source symbol plus one parity bit forms the encoded symbol, which is fed into an 8PSK mapper. An AWGN channel is assumed in the simulation. Fig. 3 shows the simulation comparison of the proposed simplified SOVA to the MAP algorithm for TTCM with block length $N = 200$ and $N = 1024$ for 8 iterations. It is shown that for 8 iterations, the performance of the proposed SOVA is within 0.2dB of the Log-MAP at BER=10^{-4}.

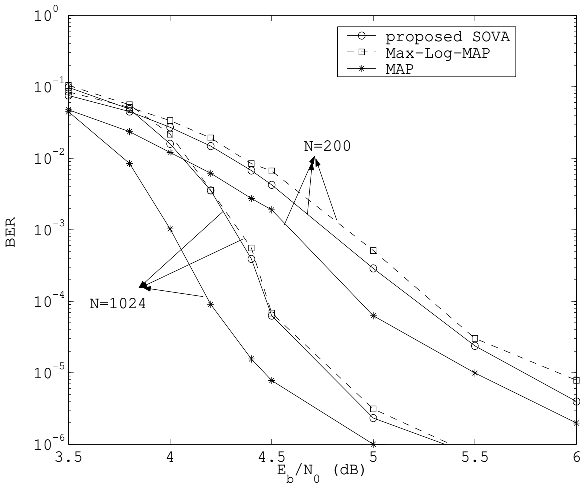

Fig. 3. Proposed SOVA and Log-MAP for 8 state, 8-PSK TTCM, with decoder 8 iterations

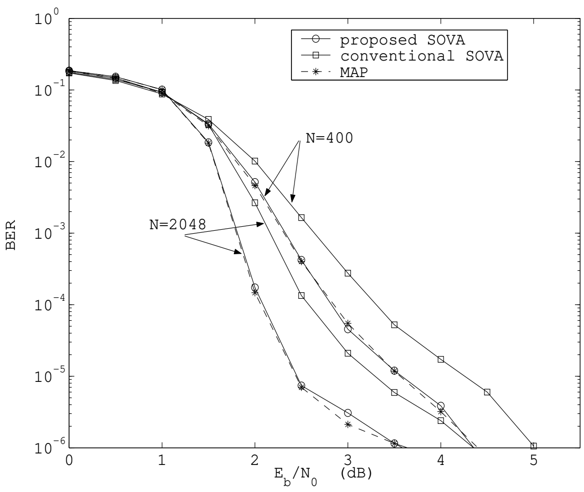

Fig. 4. BER comparison for binary Turbo decoding with 8 iterations

For the binary Turbo codes simulation, the component encoder is a systematic recursive rate-1/2 convoluational encoder with generator polynomials $g_0(D) = 1 + D + D^2$ and $g_1(D) = 1 + D^2$. The simulation results of the proposed SOVA, together with the conventional SOVA and MAP algorithm are shown in Fig. 4 for 400-bit and 2048-bit block length. It is concluded that the performance of the proposed SOVA is better than the conventional SOVA (about 0.5dB at BER$= 10^{-4}$, 8th iteration) and it performs very close to the MAP algorithm.

VI. Conclusions

An optimal MAP-equivalent SOVA scheme, with a sub-optimal simplified algorithm, for non-binary codes is proposed in this paper. The proposed SOVA is different from the conventional SOVA in the literature. It can be also applied to the binary codes. The simulation results for Turbo codes show that its suboptimal algorithm performs better than the conventional SOVA and close to that of optimal Log-MAP algorithm. It is shown that the proposed SOVA is a proper detecting/decoding algorithm not only for bandwidth efficient TTCM, but also for binary Turbo codes.

References

[1] G. D. Forney, "The Viterbi Algorithm," *Proc. Of the IEEE*, Vol. 61, No. 3, pp. 268–278, Mar. 1973.

[2] 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.

[3] G.Battail, "Pondération des symboles décodés par l'algorithme de Viterbi," *Ann. Télécommun.,* Vol.42, pp.31-38, Jan. 1987.

[4] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *IEEE GLOBECOM '89*, Dallas, Texas, pp. 47.1.1–47.1.7, Nov. 1989.

[5] C. Berrou, A. Glavieux and P. Thitimajhima,"Near Shannon limit error correcting coding and decoding: turbo-codes," in *IEEE Int. Conf. on Commun.*, pp. 1064–1070, May 1993.

[6] J. Hagenauer and L. Papke, "Decoding turbo codes with the soft output Viterbi algorithm (SOVA)," *IEEE Int. Symp. on Info. Theory*, p. 164, 1994.

[7] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. on Info. Theory*, Vol. 41, No. 3, pp. 704–713, May 1995.

[8] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *IEEE Int. Conf. on Communications*(Seattle, WA, June 1995), pp. 1009–1013.

[9] J. Hagenauer, "Source-Controlled Channel Decoding," *IEEE Trans. on Commun.*, Vol. 43, No. 9, pp. 2449–57, September 1995.

[10] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Algorithm for continuous decoding of turbo codes," *Electronics Letters*, Vol. 32, No. 4, pp. 314–315, Feb. 15 1996.

[11] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Info. Theory*, Vol 42, No. 2, pp. 429–445, March 1996.

[12] P. Robertson and T. Worz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE J. on Sel. Area in Comm.* , Vol. 16, No. 2, pp. 206–218, February 1998.

[13] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. on Sel. Areas in Comm.* , Vol. 16, No. 2, pp. 260–264, Feb. 1998.

[14] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," *IEEE Comm. Letters*, Vol. 2, No. 5, pp. 137–139, May 1998.

[15] Ling Cong, Wu Xiaofu and Yi Xiaoxin, "On SOVA for nonbinary codes," *IEEE Comm. Letters*, Vol. 3, No. 12, pp. 335–337, Dec. 1999.