Analysis of the SISO Decoding Algorithms

Georgi V. Hristov[1]

Abstract – **In this paper we provides a synthetic introduction to the methods and fundamental principles involved in turbo-coding and in the associated iterative decoding strategy. The fundamental concepts of soft decision and of soft decoding of a binary code are therefore first considered. The recent interpretation of the turbo decoding algorithm using the mathematical framework provided by factor graph theory is then briefly explained.**

Keywords – **SISO decoding algorithms, Viterbi algorithm**

I. INTRODUCTION

There are a great variety of decoding algorithms, some heuristic and some derived from well-defined optimality criteria. The purpose of the a posteriori probability (APP) algorithm is to compute a posteriori probabilities on either the information bits or the encoded symbols. Maximizing the a posteriori probabilities by themselves leads to only minor improvements in terms of bit error rates compared to the Viterbi algorithm [1]. The algorithm was originally invented by Bahl, Cocke, Jelinek, and Raviv [2] and was used to maximize the probability of each symbol being correct, referred to as the maximum a posteriori probability (MAP) algorithm.

With the invention of turbo codes in 1993 [3], however, the situation turned, and the APP became the major representative of the so-called soft-in soft-out (SISO) algorithms for providing probability information on the symbols of the code. These probabilities are required for iterative decoding and concatenated coding schemes with soft decision decoding of the inner code, such as iterative decoding of turbo codes [4,5].

II. FUNDAMENTALS OF SISO ALGORITHMS

Markov process and convolutional codes

A Markov process may be characterized at each instant i (i = 0,…,N) by a state e_i which belongs to a finite set ε of possible states. It has the following fundamental property:

$$P\left(e_i \middle| e_{i-1},...,e_0\right) = P\left(\left(e_i \middle| e_{i-1}\right)\right) \tag{1}$$

The probability for such a process to be in a given state at instant i only depends on its state at the preceding instant i-1. Such a Markov process associates an output sequence $\underline{x}$ with an input sequence $\underline{u}$. At instant i, the entry $\underline{u}_i$ (i=1,…,N) of the sequence $\underline{u}$ in input causes a transition between the states $e_{i-1} = e'$ and $e_i = e$ of the process. It generates the corresponding symbols $\underline{x}_i$ of the output sequence $\underline{x}$.

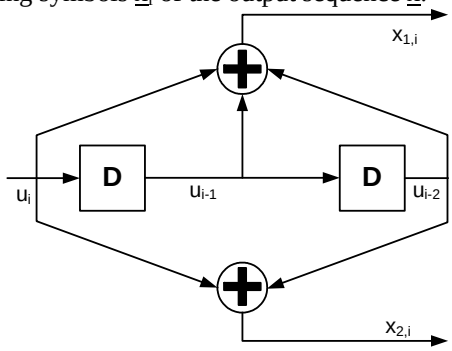

Fig.1 Rate ½ NSC code

A convolutional encoder may obviously be regarded as a Markov process, the classical case of a rate r=1/n non systematic convolutional (NSC) code is shown on Fig.1.

Assuming a code with memory M, the state of the associated Markov process at time i is then simply given by:

$$e_i = \left(u_i,...,u_{i-M+1}\right) \tag{2}$$

There are thus 2^M possible states, then it is possible to represent a convolutional code by means of a state diagram as illustrated in Fig. 2 [6].

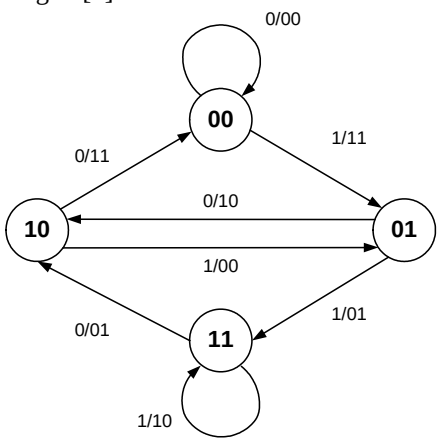

Fig.2 Code state diagram

The code trellis, which gives the code states as a function of the time index [6] is shown on Fig. 3. The encoding of a given information sequence can be associated with a path through the latter diagrams.

The problem of the decoding of a convolutional code can be seen as the problem of the estimation of the state sequence $\underline{e} = \left(e_0,...,e_N\right)$ of the associated Markov process, based on the received sequence $\underline{y}$ (which corresponds to the noisy observation of the coded sequence $\underline{x}$).

[1]Georgi V. Hristov is with the Department of Communication Systems and Technologies, 8 Studentska Str., 7017 Rousse, Bulgaria, E-mail: ghristov@mbox.contact.bg

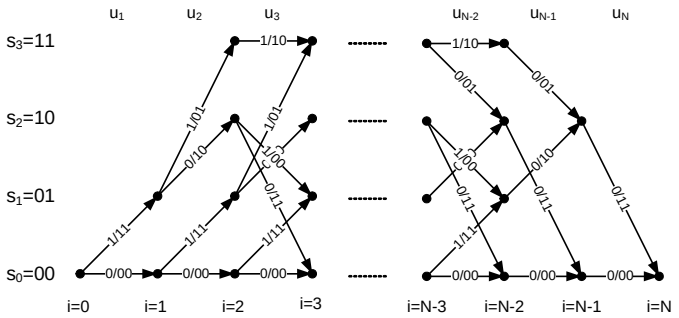

Fig.3 Code trellis diagram

Once the sequence $\hat{\underline{e}}$ is known, there is a one-to-one relation with the sequences $\hat{\underline{x}}$ and $\hat{\underline{u}}$.

Conventional decoding: MAP sequence estimation

A conventional approach for the decoding of a convolutional code is to perform MAP sequence estimation. Given the sequence $\underline{y}$, this consists in finding the information sequence $\underline{u}$ for which the a posteriori probability $P(\underline{u}|\underline{y})$ is maximum:

$$\hat{\underline{u}} = \arg\max_{\underline{u}} \{P(\underline{u}|\underline{y})\} \tag{3}$$

If no a priori information is available, this approach reduces to the so-called maximum likelihood sequence estimation (MLSE) approach:

$$\hat{\underline{u}} = \arg\max_{\underline{u}} \{P(\underline{y}|\underline{u})\} \tag{4}$$

The above problem of MAP sequence estimation can be stated more generally as follows: given the sequence $\underline{y}$ of observations of a discrete-time finite-state Markov process in memoryless noise, find the state sequence $\hat{\underline{e}}$ for which the corresponding a posteriori probability $P(\hat{\underline{e}}|\underline{y})$ is maximum:

$$\hat{\underline{e}} = \arg\max_{\underline{e}} \{P(\underline{e}|\underline{y})\} \tag{5}$$

This is obviously equivalent to:

$$\hat{\underline{e}} = \arg\max_{\underline{e}} \{P(\underline{y}|\underline{e})\} \tag{6}$$

Due to the Markov and memoryless properties of the system, we can decompose $P(\underline{e}|\underline{y})$ on the following way:

$$P(\underline{e}|\underline{y}) = \prod_{i=1}^{N} P(e_i = e|e_{i-1} = e') \prod_{i=1}^{N} P(y_i|e_i = e, e_{i-1} = e') \tag{7}$$

The above path metric in the logarithmic domain can be decomposed into a sum of individual transition metrics:

$$\ln(P(\underline{e}, \underline{y})) = \sum_{i=1}^{N} \bar{\gamma}_i (e_{i-1} = e', e_i = e) \tag{8}$$

III. Symbol-by-Symbol APP Estimation

The associated SISO decoder has to provide the a posteriori LLR sequence $\underline{L}_p$ with entries:

$$L_p(u_i) = \ln\left(\frac{P(u_i = 1|\underline{y})}{P(u_i = 0|\underline{y})}\right) \tag{9}$$

on the basis of the received sequence $\underline{y}$, and of the a priori information sequence $\underline{L}_a$ with entries $L_a(u_i) = \ln\frac{P(u_i = 1)}{P(u_i = 0)}$.

The detected bits are finally found using hard detection:

$$\hat{u}_i = \begin{cases} 1 & if \;\; L_p(u_i) \geq 0 \\ 0 & if \;\; L_p(u_i) < 0 \end{cases} \tag{10}$$

This decoding strategy is equivalent to finding the most likely information symbols u_i (i=1,…,N) given the observed coded sequence $\underline{y}$:

$$\hat{u}_i = \arg\max_{u_i} P(u_i|\underline{y}), \; (i=1,\ldots,N) \tag{11}$$

The a posteriori probabilities of the states and transitions of a Markov source observed through a discrete-time memoryless channel is optimally solved by the BCJR algorithm [1]. Based on the latter algorithm, a slight modification enables to provide the symbol a APP's $P_p(u_i) \overset{\Delta}{=} P(u_i|\underline{y})$, and by the way to form the required a posteriori LLR's at the SISO decoder output. [1, 6, 7].

The MAP (BCJR) algorithm

We considered a binary coded communication system, assuming a rate r=1/n NSC code. The available data is the sequence $\underline{y}$ of the received symbols, the sequence $\underline{L}_a$ of a priori information, the initial state $\in_0$ and the final state $\in_N$ of the encoding process, the code trellis, and the channel characteristics.

Considering a given transition in the trellis at time i, the a posteriori LLR's given in (9) is:

$$L_p(u_i) = \ln\left(\frac{\sum \varepsilon^+ p(e_{i-1} = e', e_i = e|\underline{y})}{\sum \varepsilon^- p(e_{i-1} = e', e_i = e|\underline{y})}\right) \tag{12}$$

where ε^+ (resp. ε^-) is the set of transitions ($e_{i-1} = e'$, $e_i = e$) caused by a symbol u_i=1 (resp. u_i=0). This can be simplified as:

$$L_p(u_i) = \ln\left(\frac{\sum \varepsilon^+ p(e_{i-1} = e', e_i = e, \underline{y})}{\sum \varepsilon^- p(e_{i-1} = e', e_i = e, \underline{y})}\right) \quad (13)$$

The problem is to evaluate the probability $p(e_{i-1} = e', e_i = e, \underline{y})$. The latter probability can be decomposed as:

$$p(e_{i-1} = e', e_i = e, \underline{y}) = \alpha_{i-1}(e').\gamma_i(e', e).\beta_i(e) \quad (14)$$

These quantities can be respectively evaluated as follows:

1. The parameters α are obtained via the following recursion:

$$\alpha_i(e) = \sum_{e' \in \varepsilon} \alpha_{i-1}(e').\gamma_i(e', e), \ (i = 0, ..., N-1; \forall e \in \varepsilon) \quad (15)$$

It is called forward recursion, as it implies to go through the trellis from the state e_0 till the state e_N. The following initial conditions are used for the recursion:

$$\alpha_0(e_0) = 1 \text{ and } \alpha_0(e \neq e_0) = 0 \quad (16)$$

which means that the coder is assumed to begin in the state e_0.

2. The parameters β are obtained in practice via the following recursion:

$$\beta_{i-1}(e') = \sum_{e \in \varepsilon} \beta_i(e).\gamma_i(e', e), \ (i = 2, ..., N+1; \forall e' \in \varepsilon) \quad (17)$$

It is called backward recursion, as it implies to go through the trellis from the state e_N to the state e_0. If trellis termination is implemented at the encoder, the following initial conditions are used:

$$\beta_N(e_N) = 1 \text{ and } \beta_N(e \neq e_N) = 0 \quad (18)$$

which means that the encoder is assumed to end in the state e_N. If no trellis termination is implemented, the following initial conditions are then used:

$$\beta_N(s) = \frac{1}{2^M}, \ \forall e \in \varepsilon \quad (19)$$

which means that one may end with the same probability in each of the 2^M possible states.

3. The parameter $\gamma_i(e', e)$ is associated with a transition between the states $e_{i-1} = e'$ and $e_i = e$. We have that:

$$\gamma_i(e', e) = p(\underline{y}_i | e_{i-1} = e', e_i = e).P(e_i = e | e_{i-1} = e') \quad (20)$$

Written in terms of symbols rather than in terms of states, this expression becomes:

$$\gamma_i(e', e) = p(\underline{y}_i | u_i, e_{i-1} = e').P(u_i) \quad (21)$$

The first factor $p(\underline{y}_i | u_i, e_{i-1} = e')$ is evaluated on the basis of the received symbols and of the channel type, whereas the second factor $P(u_i)$ is evaluated on the basis of the available a priori information $L_a(u_i)$. Parameter $\gamma_i(e', e)$ is often referred to as the metric associated with the transition ($e_{i-1} = e'$, $e_i = e$).

The MAP algorithms evaluates the a posteriori LLR's $L(u_i)$ of the information bits u_i (for i=1,…,N) according to:

$$L_p(u_i) = \ln\left(\frac{\sum \varepsilon^+ \alpha_{i-1}(e').\gamma_i(e', e).\beta_i(e)}{\sum \varepsilon^- \alpha_{i-1}(e').\gamma_i(e', e).\beta_i(e)}\right) \quad (22)$$

where the parameters α and β are obtained through recursions based on (15) and (17). The parameters γ are calculated according to (21), based on the received symbols, the considered channel, and the a priori information available about the transmitted information symbols.

The MAX-LOG-MAP algorithm

The MAP algorithm suffers from numerical problems related to the finite precision representation of numbers. As shown in [7], these problems can be solved by working in the logarithmic domain: $\overline{\alpha}_i(e) = \ln(\alpha_i(e))$, $\overline{\beta}_i(e) = \ln(\beta_i(e))$ and $\overline{\gamma}_i(e) = \ln(\gamma_i(e))$. In this case, (22) can be reformulated as:

$$\begin{aligned} L_p(u_i) = &\ln\left(\sum_{\varepsilon^+} \exp\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e', e) + \overline{\beta}_i(e)\right)\right) \\ &- \ln\left(\sum_{\varepsilon^-} \exp\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e', e) + \overline{\beta}_i(e)\right)\right) \end{aligned} \quad (22)$$

This expression can then be further simplified by using the approximation for the logarithm of a sum of exponentials:

$$\ln(\exp(x) + \exp(y) + \exp(z)) \approx \max(x, y, z) \quad (23)$$

This leads to:

$$\begin{aligned} L_p(u_i) \approx &\max_{\varepsilon^+}\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e', e) + \overline{\beta}_i(e)\right) \\ &- \max_{\varepsilon^-}\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e', e) + \overline{\beta}_i(e)\right) \end{aligned} \quad (24)$$

The parameters $\overline{\alpha}$, $\overline{\beta}$ and $\overline{\gamma}$ are calculated as follows.

1. The parameters $\overline{\alpha}$ are calculated according to the following forward recursion, with initial conditions modified accordingly:

$$\overline{\alpha}_i(e) = \max_{e' \in \varepsilon}\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e', e)\right) \quad (25)$$

2. Similarly, the parameters $\overline{\beta}$ are calculated according to the following backward recursion with initial conditions modified accordingly:

$$\overline{\beta}_{i-1}(e) = \max_{e \in \varepsilon}\left(\overline{\beta}_i(e) + \overline{\gamma}_i(e', e)\right) \quad (26)$$

3. The parameters $\overline{\gamma}$ are calculated based on (21) transposed in the logarithmic domain:

$$\overline{\gamma}_i(e',e) = \ln\left(p\left(y_i | u_i, e_{i-1} = e'\right)\right) + \ln\left(P(u_i)\right) \tag{27}$$

The LOG-MAP algorithm

It is possible to preserve the optimality of the MAP algorithm while keeping all the advantages associated with the formulation in the logarithmic domain. Instead of using the approximation given by (23), we must therefore use the following exact expression [7]:

$$\begin{aligned}\ln\left(\exp(x)+\exp(y)\right) &= \max(x,y) + \ln\left(1+\exp\left(-|x-y|\right)\right)\\ &= \overset{*}{\max}(x,y)\end{aligned} \tag{28}$$

If there is more than two entries, one has to proceed recursively:

$$\begin{aligned}\ln\left(\exp(x)+\exp(y)+\exp(z)\right) &= \overset{*}{\max}(x,y,z)\\ &= \overset{*}{\max}\left(\overset{*}{\max}(x,y),z\right)\end{aligned} \tag{29}$$

This function may be considered as a generalized maximum function. The LOG-MAP algorithm proceeds exactly the same way as the MAX-LOG-MAP algorithm, the only difference being that the classical maximum function is replaced by the above generalized maximum function in (24), (25) and (26). The a posteriori LLR write:

$$\begin{aligned}L_p(u_i) \approx &\overset{*}{\max_{\varepsilon^+}}\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e',e) + \overline{\beta}_i(e)\right)\\ &- \overset{*}{\max_{\varepsilon^-}}\left(\overline{\alpha}_{i-1}(e') + \overline{\gamma}_i(e',e) + \overline{\beta}_i(e)\right)\end{aligned} \tag{30}$$

The obtained algorithm is equivalent to the optimal MAP algorithm, but enables to avoid the associated numerical problems. The complexity of this algorithm evolves in $O(2^K)$, where $K=M+1$ is the length of the considered Markov process [7].

IV. Simulation Result

In Fig. 4 simulation results are provided at iterations 1 to 5, considering a turbo-code whose elementary SISO decoders are implemented via the suboptimal MAX-LOG-MAP algorithm (dotted line) and via the LOG-MAP algorithm (in continuous). The following configuration is considered: interleaver of length N=1024, code rate r=1/2, 2 identical RSC encoders of length K=5 and with generator polynomials ([11111], [10001]), BPSK modulation and AWGN channel. As it could be expected, the LOG-MAP algorithm outperforms the suboptimal MAX-LOG-MAP algorithm. At a BER of 10^{-4}, a difference of 0,5 dB can be observed at iteration 5.

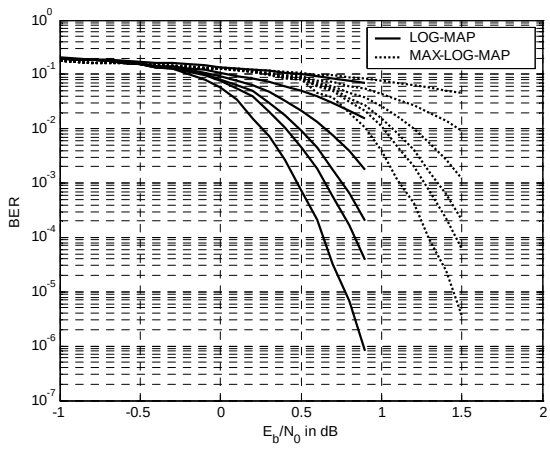

Fig.4 Performance of SISO decoding algorithms

V. Conclusion

The simulation result shows that at low E_b/N_0 ratios, the MAX-LOG-MAP algorithm leads to noticeable performance degradation. At high E_b/N_0 ratios and for a sufficient iteration number, the MAX-LOG-MAP algorithm offers performance quasi-identical to those of the LOG-MAP algorithm, because the approximation on which the MAX-LOG-MAP algorithm is based becomes more valuable as the considered signal-to-noise ratio is increasing.

Acknowledgement

This work is a part of the research project ВУ-ТН-105/2005 of Bulgarian Science Fund at Ministry of Education and Science.

References

[1] A. J. Viterbi, “An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes", IEEE Journal on Selected Areas in Communications, vol. 16, no. 2, pp. 260-264, 1998.

[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, 1974.

[3] C. Berrou, A. Glavieux, P. Thitmajshima, “Near Shannon limit error – correction coding and decoding: turbo–codes”, in Proc. IEEE International Conf. on Commun. (ICC), Geneva, Switzerland, pp.1064–1071, 1993.

[4] T. Iliev, “A study of turbo codes for UMTS third generation cellular standard”, CompSysTech’2007, Conference Proceedings, pp. VI.6 – VI.7, Ruse, Bulgaria, 2007

[5] T. Iliev and D. Radev, “Turbo codes in the CCSDS standard for wireless data”, ICEST’2007, Conference Proceeding, pp. 31 – 35, Ohrid, Macedonia, 2007

[6] J. Hagenauer, E. Offer, L. Papke, “Iterative decoding of binary block and convolutional codes", IEEE Transactions on Information Theory, vol. 42, pp. 429-445, 1996.

[7] P. Robertson, P. Hoeher, E. Villebrun, “A comparison of optimal and sub-optimal MAP decoding algorithms operating in the logdomain",- IEEE International Conference on Communications, vol. 2, pp. 1009-1013, 1999.