Complexity Reduction of Turbo Decoding Based on Decision Thresholds

Luís O. Mataveli and Celso de Almeida

***Abstract*—In this paper we propose a method for reducing turbo decoding complexity. Based on decision thresholds, some calculations are previously considered negligible, and it is possible to reduce the number of calculations required to perform turbo decoding. Simulations show that almost 40% of the state metrics (α and β) and 25% of the LLRs (Log-Likelihood Ratios) calculations can be eliminated. With the same idea, decision thresholds can be used to take decisions on the information bits based on the LLRs, achieving good decoding complexity reduction, especially for high SNR (Signal to Noise Ratio).**

***Index Terms*—Turbo decoding, complexity reduction, decision thresholds.**

I. Introduction

Turbo codes [1] are widely used in wireless communication systems with the purpose of error correction. They are used in satellite communications, WiMAX, LTE standards, etc., and have high decoding complexity. Currently, smartphones, for example, have to support multiple radio standards, media and graphics applications, which results in a charge of about 10^{11} operations per second limited to a power of 1 Watt [2]. More than 50% of the operations are performed in the radio processing layer (demodulation and decoding). The processing required for decoding using turbo codes, is much higher when compared to other processes such as equalization, channel estimation, interference cancellation and timing [3]. So it is important to reduce the number of operations in turbo decoding.

There are some papers that propose different methods for reducing turbo decoding complexity. For example, [4] proposes a method where the LLRs in the current and in the previous iterations are compared. If LLRs have similar values, it takes a decision on the information bit. In the next iteration, the trellis section associated with the decided bit is discarded, reducing the number of calculations. However, to use this method it is necessary to perform at least two complete iterations.

Another method, proposed in [5], takes decisions based on the LLRs using a decision threshold. The branches corresponding to the decided bits are removed from that trellis section and the survivor branches are connected to the branches in the neighboring sections. This method achieves a good complexity reduction, but it has a waste of time of about 6% due to the processing required to connect the branches of the trellis. Besides these two methods, there are others in the literature [6], [7], [8].

In this paper, we propose a new method of complexity reduction that consists in first removing the "weak" branches of the trellis, i.e. those branches that have small metrics (α, β and γ). Through simulations, we note that these branches have negligible metrics when calculating the LLR. The branches are classified as "weak" based on decision thresholds. Another threshold is defined that take decisions on the information bits based on LLRs, which avoids the calculation of more iterations when it is not necessary.

The performance in terms of bit error rate (BER) and complexity are obtained as a function of the ratio E_b/N_0 using the Monte Carlo simulation method.

In Section II we analyze the complexity of turbo decoding and in Section III we describe the proposed method to reduce the complexity. The results are shown in Section IV and the conclusions in Section V.

II. Turbo decoding

In this paper we consider a turbo code formed by two recursive systematic convolutional codes of rate $r_c = 1/2$, concatenated in parallel and separated by a random interleaver. Puncturing is used, resulting in a turbo code of rate $r_t = 1/2$. For turbo decoding it is considered the MAP (*Maximum A Posteriori*) algorithm, but we believe that the complexity reduction technique proposed in this paper can also be applied to Log-MAP and Max-Log-MAP algorithms.

The MAP algorithm [9] provides a number that is the probability *a posteriori* of the information bit b_k be 1 or -1. This probability is the LLR (*Log Likelihood Ratio*) which it is calculated by the logarithm of the ratio between the *a posteriori* probability of a particular bit be 1 and -1. The LLR is given by:

$$L\left(b_k|\mathbf{y}\right) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ b_k=+1}} \alpha_{k-1}\left(s'\right).\gamma_k\left(s',s\right).\beta_k\left(s\right)}{\sum_{\substack{(s',s)\Rightarrow \\ b_k=-1}} \alpha_{k-1}\left(s'\right).\gamma_k\left(s',s\right).\beta_k\left(s\right)}\right) \quad (1)$$

where $(s',s) \Rightarrow b_k = +1$ represents the transitions from state s' to state s, corresponding to information bit $b_k = +1$ and, in the same way, $(s',s) \Rightarrow b_k = -1$ is the transitions corresponding to information bit $b_k = -1$. State metrics (α and β) are calculated recursively as illustrated in Fig. 1 and are expressed as:

$$\alpha_k\left(s\right) = \sum_{s'} \alpha_{k-1}\left(s'\right).\gamma_k\left(s',s\right) \quad (2)$$

Luís O. Mataveli and Celso de Almeida, Department of Communications, Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas-SP, Brazil, E-mails: mataveli@decom.fee.unicamp.br, celso@decom.fee.unicamp.br.

and

$$\beta_k\left(s'\right) = \sum_s \beta_{k+1}\left(s\right).\gamma_{k+1}\left(s', s\right), \quad (3)$$

where k is the trellis section identifier and the branch metrics are given by:

$$\gamma_k\left(s', s\right) = \frac{\exp\left(L_e/2\right)}{1+\exp\left(L_e\right)} \exp\left[\frac{1}{2}\left(x_s L_e + L_c x_s y_s + L_c x_p y_p\right)\right], \quad (4)$$

where L_e is the extrinsic information that is just calculated by the previous decoder for the information bits, y_s and y_p are the received samples corresponding to the systematic and parity bits, respectively. x_s and x_p are the encoded systematic and parity bits, respectively, associated to each branch of the trellis. L_c is the channel reliability, that is given by $L_c = 4 \cdot r_t \cdot {}^{E_b}/_{N_0}$, where ${}^{E_b}/_{N_0}$ is the ratio between bit energy and noise power spectral density.

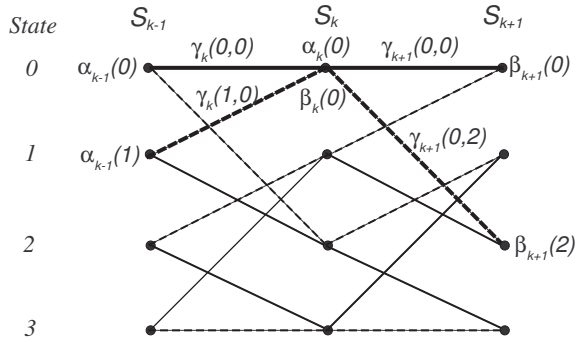

Figure 1. Recursive calculation of $\alpha_k\left(s\right)$ and $\beta_k\left(s'\right)$.

We can measure the turbo decoding complexity by the number of operations necessary to calculate (1) to (4), that is proportional to the number of branches in the trellis, $n = 2^{i+m_t}$, where i is the number of input bits and m_t is the number of memories in the constituent convolutional encoders. Calculations are made for $\gamma_k\left(s', s\right)$ in all branches in the trellis, for $\alpha_k\left(s\right)$ and $\beta_k\left(s'\right)$ in all states and for LLR in each trellis section. These calculations are made in each constituent decoder and repeated in all iterations.

The proposed method aims at eliminating some of these calculations based on decision thresholds, as described in the next section.

III. Method for reducing turbo decoding complexity

As seen in the previous section, the turbo decoding using MAP algorithm requires many calculations. In this section we present a method that eliminates some calculations of turbo decoding.

The proposed method consists in eliminating the "weak" branches based on decision thresholds. First, we set two thresholds, one for $\alpha_{k-1}\left(s'\right)$ and $\beta_k\left(s\right)$ named $L_{\alpha\beta}$, and another for $\gamma_k\left(s', s\right)$ named L_γ. With these thresholds, the following rule is used for the state metrics calculation:

- If $\left(\alpha_{k-1}\left(s'\right) \geq L_{\alpha\beta}\right)$ or if $\left(\gamma_k\left(s', s\right) \geq L_\gamma\right)$;
- Then $\alpha_k\left(s\right)$ is calculated.
- Else, the branch is discarded.

Thus, a branch must be considered in the calculation of $\alpha_k\left(s\right)$ when $\alpha_{k-1}\left(s'\right)$ or $\gamma_k\left(s', s\right)$ have high values. When both are small, the branch is classified as "weak" and it is discarded prior to performing the calculations.

Similarly, we have the rule for $\beta_k\left(s'\right)$ calculation:

- If $\left(\beta_{k+1}\left(s\right) \geq L_{\alpha\beta}\right)$ or if $\left(\gamma_{k+1}\left(s', s\right) \geq L_\gamma\right)$;
- Then $\beta_k\left(s'\right)$ is calculated.
- Else, the branch is discarded.

The calculations of $\alpha_k\left(s\right)$ and $\beta_k\left(s'\right)$ consider all branches reaching each state in the trellis, as shown in (2) and (3). The proposed method avoids some of these calculations, because based on decision thresholds it is possible to know that the state metric contribution is very small and therefore can be discarded. For example, consider that $\alpha_{k-1}\left(1\right)$ and $\gamma_k\left(1, 0\right)$, in Fig. 1, are smaller than $L_{\alpha\beta}$ and L_γ, respectively. Thus, these terms are discarded and the computation of $\alpha_k\left(0\right) = \alpha_{k-1}\left(0\right) \cdot \gamma_k\left(0, 0\right) + \alpha_{k-1}\left(1\right) \cdot \gamma_k\left(1, 0\right)$ is done by $\alpha_k\left(0\right) = \alpha_{k-1}\left(0\right) \cdot \gamma_k\left(0, 0\right)$. In the LLR calculation in (1), if any $\alpha_{k-1}\left(s'\right)$ or $\beta_k\left(s\right)$ is not calculated, then the multiplication $\alpha_{k-1}\left(s'\right) \cdot \gamma_k\left(s', s\right) \cdot \beta_k\left(s\right)$ is not calculated too. To use the threshold L_γ, in this method we calculate all the branch metrics of the trellis.

The second part of the proposed method consists in defining a threshold for LLRs, named L_{llr}. With this threshold, we set a decision criterion to previously decide the information bits and stop decoding:

- If $|LLR_k| > L_{llr}$ for all bits in a block, i.e., for $1 \leq k \leq N$, then stops decoding.

where N is the information bits block length.

The second part of the proposed method consists in defining a reliability threshold for the LLRs. That is, if the absolute value of LLR is large, then we believe that it will not change its signal in the subsequent iterations, keeping the same decision on the information bit. For example, if $LLR_k = 15$ and $L_{llr} = 10$ then we can decide $\hat{b}_k = 1$, because probably LLR will not change the signal in subsequent iterations. When all LLRs in a block are reliable, i.e., their absolute values are greater than the threshold L_{llr}, then the decoding process ends and it is not necessary to calculate any additional iteration, reducing the decoding complexity. The results presented in Section IV show that with the threshold $L_{llr} = 10$ it is possible to reduce decoding complexity without any performance degradation.

IV. RESULTS

The results are obtained by simulation using the Monte Carlo method. The turbo code consists of two recursive systematic convolutional codes (RSC) with matrix generator $G = [7, 5]$ in octal form, and separated by a random interleaver. We consider puncturing resulting in a turbo code rate $r_t = 1/2$. The information bits are transmitted in blocks of 1000 bits and we analyze performance and complexity by considering an AWGN channel.

System performance is evaluated in terms of bit error rate (BER) as a function of E_b/N_0. We consider that the complexity is given by the number of calculations to perform turbo decoding using the proposed method normalized by the number of calculations to perform the original MAP algorithm.

In simulations, we have observed that the thresholds $L_{\alpha\beta} = 0.25$ and $L_\gamma = 10^{-3}$ achieve good complexity reduction without any performance degradation. Thus, at first we analyze the performance and complexity considering these thresholds. After that, we also consider a threshold for the LLRs.

Fig. 2 shows the turbo code performance considering only thresholds $L_{\alpha\beta}$ and L_γ for different number of iterations. As a reference, the dashed line shows the turbo code performance using original MAP algorithm after 8 iterations. Comparing to the proposed method after 8 iterations with original MAP algorithm, we can see that both have almost the same performance. For E_b/N_0 greater than 2.5 dB, the decoding performance using the proposed method is slightly better than the original MAP algorithm, but this is due to imprecision in simulations.

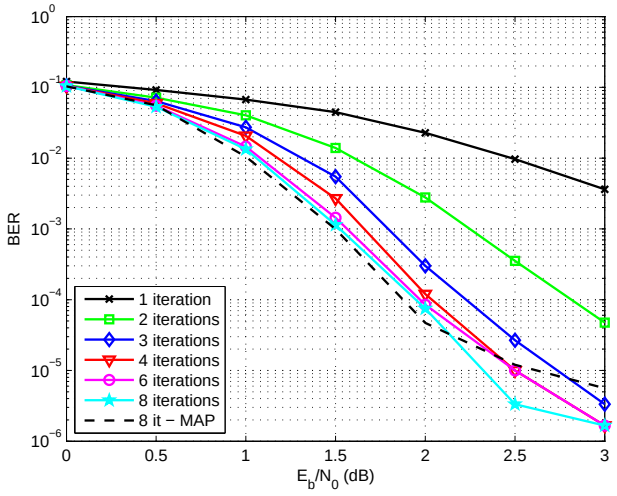

Figure 2. Reduced complexity turbo decoding performance for $L_{\alpha\beta} = 0.25$ and $L_\gamma = 10^{-3}$.

Fig. 3 shows the normalized complexity as a function of E_b/N_0 and the number of iterations. The complexities of α and β are given by the number of calculations in (2) and (3) and are shown in the first and second graphs. Complexity of LLR is measured by the number of calculations of $\alpha_{k-1}(s').\gamma_k(s',s).\beta_k(s)$ in the numerator and denominator of (1), and is shown in the third graph. We can see, for example for $E_b/N_0 = 3$ dB, that the complexity of α and β is 95% and 70% in the first and second iterations, respectively. In the third iteration, the complexity is about 63%, i.e. 37% of computation reduction. The complexity to calculate the LLRs drops to 75% in the third iteration. Fig. 2 and 3 show that it is possible to eliminate many calculations in turbo decoding without performance degradation.

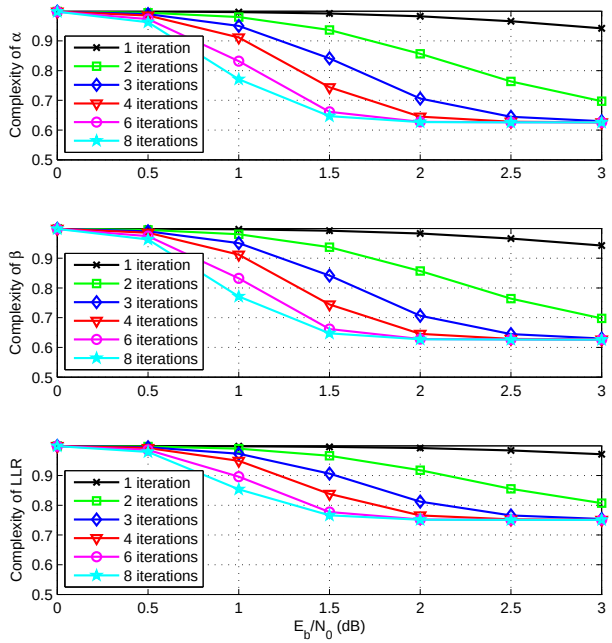

Figure 3. Normalized complexity of turbo decoding for $L_{\alpha\beta} = 0.25$ and $L_\gamma = 10^{-3}$.

We can also significantly reduce the number of calculations when we define a threshold for the LLRs. After each iteration the absolute value of all LLRs is compared to the threshold $L_{llr} = 10$. If all values in a block are higher than the threshold, then the decoding stops and a decision is taken to decide all information bits. Thus, it is not necessary to do more iterations, if the LLRs are reliable. Fig. 4 shows the performance and Fig 5 shows the complexity of α, β and LLRs. When no iteration is performed, complexity is considered to be zero. We can see that for $E_b/N_0 = 3$ dB, it is not necessary to do more calculations from the fourth iteration. In the first iteration the turbo decoder still needs to do about 95% of the calculations, which is expected because it is still early to take a decision on the information bits. In the second iteration, the decoder does only about 70% of α and β and 80% of LLRs calculations. In the third iteration the number of calculations decreases to 25% for α and β and 30% for LLR. In the fourth and higher iterations, the complexity is almost zero.

Fig. 5 shows that the decoding complexity decreases when

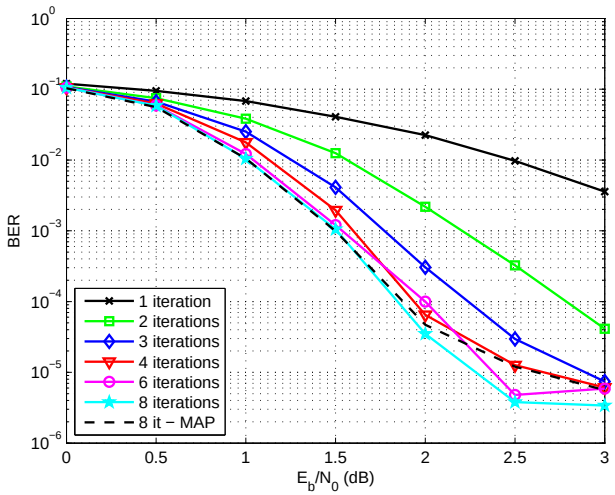

Figure 4. Reduced complexity turbo decoding performance for $L_{\alpha\beta} = 0.25$, $L_\gamma = 10^{-3}$ and $L_{llr} = 10$.

E_b/N_0 increases. For higher values of E_b/N_0, for example 10 dB, the simulations showed that it is only necessary to calculate the first iteration when $L_{llr} = 10$.

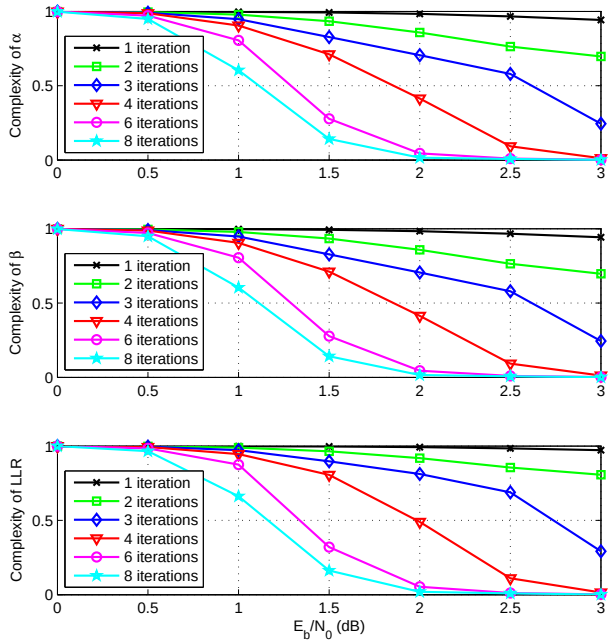

Figure 5. Normalized complexity of turbo decoding for $L_{\alpha\beta} = 0.25$, $L_\gamma = 10^{-3}$ and $L_{llr} = 10$.

V. Conclusions

We have proposed a method for reducing the turbo decoding complexity based on decision thresholds. The performance and complexity using this method were compared with the MAP algorithm and we have concluded that it is possible to have the same performance in terms of BER, but with lesser complexity. Complexity is measured by the number of operations required to calculate the state metrics (α and β) and LLRs.

The complexity reduction varies with E_b/N_0 and with the number of iterations. For $E_b/N_0 = 3$ dB the complexity reduction was 37% for α and β calculations, and 25% for LLR for 3 or more iterations. When we have also considered a threshold for LLRs, the complexity goes to zero after the fourth iteration.

When the thresholds $L_{\alpha\beta}$ and L_γ decrease, most metric states have to be calculated, increasing the turbo decoding complexity, but improving BER. Otherwise, as thresholds increase more metrics are ignored in the calculations, which reduces the decoding complexity, but increases the BER. According to simulations, with thresholds higher than the proposed ones, the complexity decreases, but BER increases considerably.

Regarding the threshold for LLRs, the complexity increases when L_{llr} also increases, because fewer LLRs will be smaller than the threshold. When L_{llr} decreases, the complexity also decreases, but BER increases. We showed that with a threshold $L_{llr} = 10$ it is possible to achieve a good trade off between complexity reduction and performance in terms of BER.

Hence, we conclude that the proposed method is efficient, since it is possible to reduce the turbo decoding complexity without any significant performance degradation.

References

[1] C. Berrou, A. Glavieux and P. Thitimajshima. "Near Shannon limit error-correcting coding and decoding: turbo Codes," *Proc. International Conference on Communications*, pp. 1064-1073, 1993.

[2] C. H. van Berkel, "Multi-core for mobile phones," *Proc. Design, Automations. Test Europe Conf. Exhibition*, pp. 1260-1265, April 2009.

[3] F. Kienle, N. Wehn and H. Meyr, "On Complexity, Energy- and Implementation-Efficiency of Channel Decoders," *IEEE Transactions on Communications*, v.99, pp. 3301-3310, December 2011.

[4] Wu, Jinhong, Vojcic, Branimir R. and Wang, Zhengdao. "Turbo Decoding Complexity Reduction by Symbol Selection and Partial Iterations," *IEEE GLOBECOM Proceedings*, 2007.

[5] Frey, Brendam J. e Kschischang, Frank R. "Early Detection and Trellis Splicing: Reduced-Complexity Iterative Decoding," *IEEE Journal on Selected Areas in Communications*, 1998.

[6] Z. Ma, W. H. Mow and P. Fan. "On the Complexity Reduction of Turbo Decoding for Wideband CDMA," *IEEE Transactions on Wireless Communications*, v. 4, pp. 353-356, March 2005.

[7] A. Shibutani, H. Suda and F. Adachi. "Complexity Reduction of Turbo Decoding," *IEEE Vehicular Technology Conference*, v. 3, pp.1570-1574, 1999.

[8] T. Minowa and H. Imai. "Reduced-complexity iterative decoding of high-rate turbo codes," *IEEE Globecom*, v. 1, pp. 193-197, 2000.

[9] L. R. Bahl, J. Cocke, F. Jelinek e J. Raviv. "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, pp. 284-287, Março 1974.