

Complexity Reduction of the Viterbi Algorithm Based on Samples Reliability

Luís O. Mataveli and Celso de Almeida

Abstract—The Viterbi algorithm is a maximum likelihood algorithm that is used for decoding convolutional codes. In order to determine the survivor path in a trellis, it is necessary to calculate the metrics of each branch. In this paper, we propose a method that reduces the number of branches in the trellis and consequently its complexity, based on the reliability of the received signal samples. The complexity of the proposed algorithm is reduced with the number of reliable samples. The proposed algorithm achieves a performance close to the Viterbi algorithm, but with lesser complexity, depending on the reliability threshold. The performance is evaluated in terms of bit error probability and complexity, obtained by simulation, for different signal to noise ratio and reliability threshold values. The results are obtained for different convolutional encoders by considering a Rayleigh fading channel.

Index Terms—Viterbi algorithm, complexity reduction, reliability threshold.

I. INTRODUCTION

Convolutional codes are commonly used in wireless communication systems to provide error correction and temporal diversity of the information bits. Viterbi algorithm is the maximum likelihood decoder that takes a decision over a sequence of bits [1]. For this, the Viterbi algorithm uses a trellis to find the path with the lowest distance in relation to the received samples. In order to calculate the survivor path, it is necessary to calculate the metric of each branch and to accumulate the metrics at each state. The branch metrics that reach a state are compared to each other and only the branch corresponding to the lowest metric is preserved. Therefore, the decoding complexity is proportional to the number of branches in the trellis.

The greater the memory order of a convolutional encoder, the better is its performance at the expense of greater complexity decoding. The number of states and branches grows exponentially with the memory order. Therefore, a compromise between complexity and performance should be made when choosing the reliability threshold.

There are some papers that have attempted to diminish the complexity of the Viterbi algorithm. For example, it is stated in [2] that the complexity can be decreased approximately by $1/3$, but only for a specific type of encoder. In that paper, if a state metric is much larger than others, then it considers that the survivor path will pass by that state, so the others can be eliminated. In [3] a reduction of complexity can also

be achieved, but only for high signal to noise ratio. In that paper, the algorithm does not find the information message, but the error vector. So the branch with the smallest metric is kept in the trellis and the others are discarded.

In this paper, we propose a different technique to reduce the complexity of convolutional decoding, but keeping the performance close to the Viterbi algorithm. The method consists in reducing the number of branches in the trellis, which reduces the number of calculations in the same proportion. The number of branches in the trellis can be decreased based on the prior decision of some bits, whose samples are considered reliable. Thus, the complexity reduction depends mainly on the reliability threshold value used and the E_b/N_0 . In this paper, we analyze the performance and complexity of convolutional codes for a Rayleigh fading channel. We consider analysis with binary modulation, but the proposed method can be applied to other kinds of modulation.

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

The paper is described as follows. Section II presents the method for reducing the complexity of the Viterbi algorithm. Section III presents the results and in Section IV the conclusions are presented.

II. METHOD FOR REDUCING THE COMPLEXITY OF VITERBI ALGORITHM

Viterbi algorithm chooses the path with the lowest metric in a trellis. It is necessary to calculate $n_r = 2^{k+m}$ metrics in each trellis section, where n_r is the number of branches in each trellis section, k is the number of information bits and m is the memory order of the encoder.

The Viterbi algorithm uses the operation add-compare-select (ACS), which is used in the state metric calculation, by considering all branches that reach a given state. The metric of each branch is calculated by:

$$v = \sum_{i=1}^n (y_i - \hat{b}_i)^2, \quad (1)$$

where y_i is the received sample amplitude, \hat{b}_i is the i th encoded bit of the branch and n is the number of encoded bits of the convolutional encoder.

If some branches are eliminated in the trellis, the calculation of (1) for those branches is not necessary, which reduces the algorithm complexity. Besides the reduction of branch

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.

metrics calculation, eliminating some branches also reduce the number of comparisons necessary to determine the survivor path in the trellis.

In this article we propose a method that eliminates some complexity by the previous decision of some bits, when their received samples are considered reliable. A sample is classified as reliable, when its amplitude lies on a reliable region, which is defined by a pair of thresholds $\pm L$, as illustrated in Fig. 1 and is given by [4]:

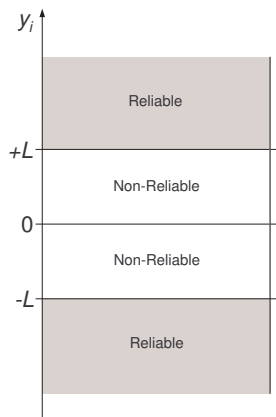
$$\begin{aligned}
 y_i &\geq +L, \text{ or} \\
 y_i &\leq -L.
 \end{aligned} \tag{2}$$


Figure 1: Reliable and non-reliable regions for the received samples.

If the sample amplitude is greater than $+L$, then we can decide by $\hat{b}_i = +1$ and if the sample amplitude is smaller than $-L$, then we can decide by $\hat{b}_i = -1$. On the other hand, if the sample amplitude is between $-L$ and $+L$, then the bit \hat{b}_i is not reliable and consequently it is not decided. For each decided bit, it is possible to eliminate some branches in the trellis. For a code with rate $r = 1/2$, for each decided bit, half of the branches is eliminated in a trellis section, reducing by half the number of calculations and consequently the complexity.

The complexity reduction depends on the ratio E_b/N_0 and on the reliability threshold value. When the threshold $L \rightarrow \infty$, the proposed method has the same performance and complexity than the Viterbi algorithm, because none sample is reliable. On the other hand, when $L = 0$ all samples are reliable and the complexity is minimal. However, the lower the threshold, the worse the decoder performance. Therefore, an appropriate value of the threshold must be chosen for the decoder to achieve a good trade off between performance and complexity.

Fig. 2 (a) shows an example of two trellis sections for an encoder with rate $1/2$ and 2 memory elements, where two encoded bits are associated to each branch. Suppose that $L = 1$, so in the first section both samples are smaller than $-L$. Consequently, the encoded bits are previously decided

as $\hat{b}_i = -1$. In this case, all branches where the encoded bits are different of $[-1 -1]$ are eliminated, reducing the number of branches to 2 and consequently the number of calculations. The eliminated branches are represented by dashed lines in Fig. 2 (b). Besides the elimination of branches based on a threshold, the proposed algorithm also eliminates disconnected branches. That is, if no branch goes into a state, the branches that goes out this state will also be eliminated, as shown in Fig. 2 (c) - (d). This reduces further the complexity.

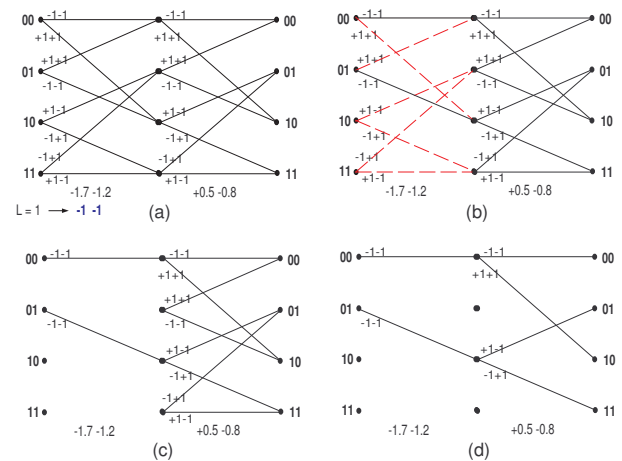


Figure 2: Trellis sections for an encoder with rate $r = 1/2$ and 4 states. (a) Original trellis. (b) Eliminated branches in dashed lines. (c) Trellis considering the eliminated branches. (d) Trellis considering the elimination of disconnected branches.

As some branches are eliminated, it is not necessary to calculate the branch metrics, because the surviving path does not pass through these branches. Moreover, it is also possible to reduce the number of comparisons, in order to determine the survivor branch which reaches each state.

We believe that the method proposed in this paper can be applied to any other type of trellis decoding, as is the case of turbo codes.

III. RESULTS

The presented results are obtained through simulations using the Monte Carlo method. We consider convolutional codes with rate $1/2$ and $1/3$ with 2 and 3 memory elements. The information bits block length is 100 bits in a non-selective Rayleigh fading channel.

The system performance is presented by curves of bit error rate (BER) as a function of E_b/N_0 for different reliability threshold values. We also present curves of complexity and time spent in simulation, where the complexity is measured by the remaining number of branches in the trellis. Both curves of complexity and simulation time are normalized with respect to the Viterbi algorithm original values.

First, we analyze the performance of a convolutional code of rate $1/2$. Fig. 3 illustrates the convolutional code performance for different reliability threshold values. As L

decreases, worst is the performance, because more previous decisions are taken, so the complexity also decreases. The encoder has memory order of 2, with matrix generator $G = [7, 5]$, in octal form.

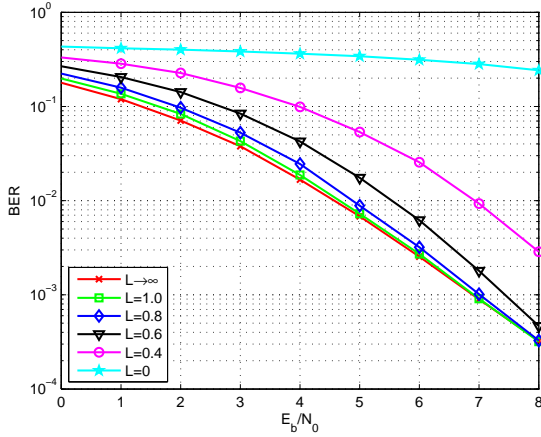


Figure 3: Performance of the convolutional encoder $G = [7, 5]$.

Fig. 4 illustrates the normalized complexity and simulation time. Observing Figs. 3 and 4, we can see that the lower is the reliability threshold L , the lower is the complexity, but the worst is the performance. When $L \rightarrow \infty$, the performance and complexity are the same of the Viterbi algorithm. For $L = 0.8$ we have almost the same performance but with a complexity of about 30% compared to the Viterbi algorithm, which means a 70% of complexity reduction. For $L = 0.6$ there is about 0.5 dB of performance degradation for a BER of 10^{-3} , but with a complexity that is only 20% of the Viterbi algorithm. For higher E_b/N_0 the performance degradation becomes negligible, even with a complexity reduction of 80%. Fig. 4 (b) shows that the simulation time decreases in equal proportion to the complexity reduction.

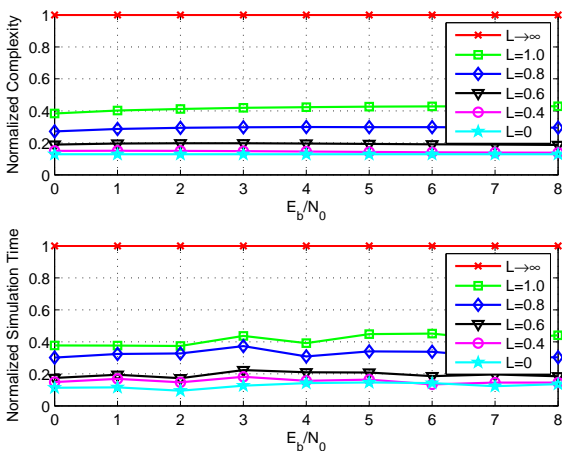
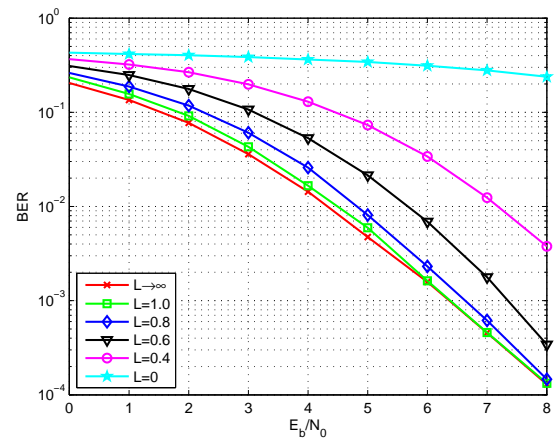
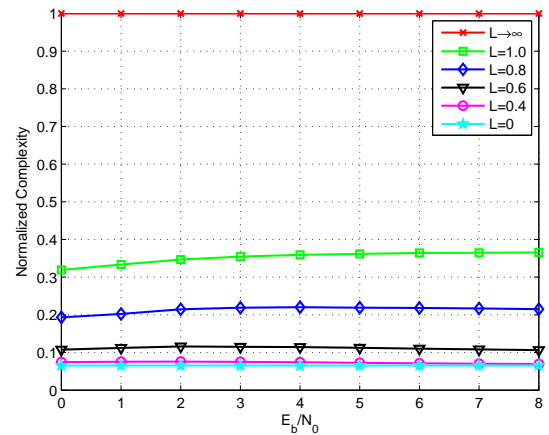


Figure 4: Normalized complexity and simulation time of the convolutional encoder $G = [7, 5]$.

Fig. 5 illustrates the performance and complexity of the encoder with generator $G = [64, 74]$, with rate $1/2$ and 3 memory elements. In this case, the complexity is measured just by the number of remaining branches in the trellis and not by the simulation time once there is an agreement between these parameters. For a threshold of 0.8, the performance is almost the same as that obtained using the Viterbi algorithm, but with a complexity reduction of about 80%. For a threshold of 0.6 it is possible to obtain a complexity reduction close to 90% with a loss in performance smaller than 1 dB.



(a) Performance.



(b) Normalized complexity.

Figure 5: Performance and normalized complexity of the convolutional encoder $G = [64, 74]$.

Comparing the two codes of rate $1/2$, we can see that the code with memory order of 3 presents greater complexity reduction, but greater degradation in performance. For example, for a threshold of 0.6 the code with 3 memory elements presents a complexity reduction of 90%, but with a loss of about 1 dB in performance, while the code with 2 memory elements has a complexity reduction of 80%, but with 0.5 dB of performance degradation.

Now we analyze the behavior of two codes of rate $r = 1/3$,

with memory order of 2 and 3. Fig. 6 illustrates the performance and normalized complexity for different reliability thresholds for the encoder $G = [5, 7, 7]$ of rate $1/3$ and 2 memory elements. We notice that for a BER higher than 10^{-4} , a reliability threshold of 1.25 is enough to achieve a performance close to the Viterbi algorithm, but with complexity of about 35%. For a threshold of 1.0 there is a loss of about 0.5 dB in performance, but with a complexity reduction of 77%.

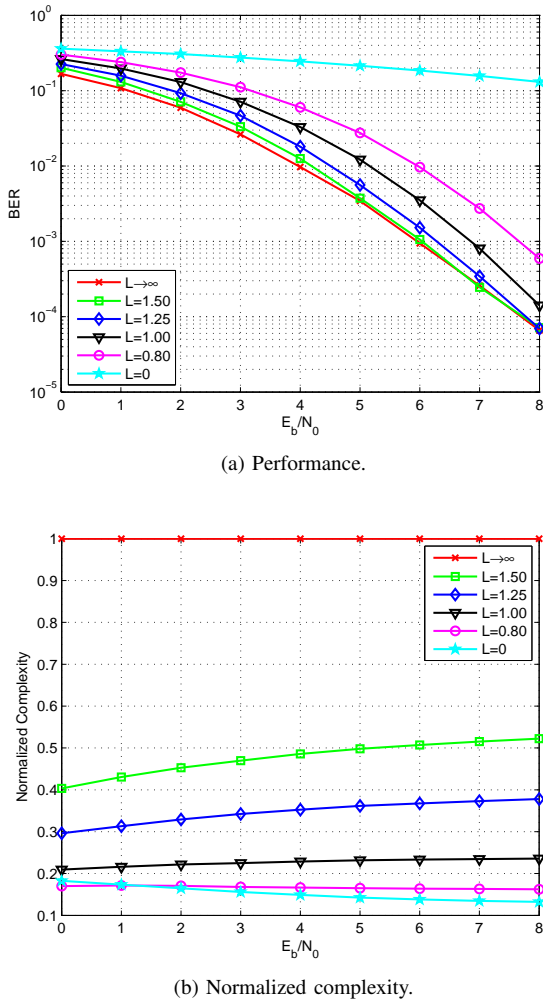


Figure 6: Performance and normalized complexity of the convolutional encoder $G = [5, 7, 7]$.

Fig. 7 shows the performance and normalized complexity for the code $G = [54, 64, 74]$, of rate $1/3$ and 3 memory elements. We can see that for a threshold of 1.75 the performance is almost the same of the Viterbi algorithm, but with a complexity ranging between 38% and 52%, depending on the E_b/N_0 value. For BER higher than 10^{-4} the performance for $L = 1.5$ tends to be similar to the Viterbi algorithm, but with complexity reduction close to 50%.

Observing Fig. 6 and 7, we can notice that both codes can reach a performance close to the Viterbi with similar complexity reduction.

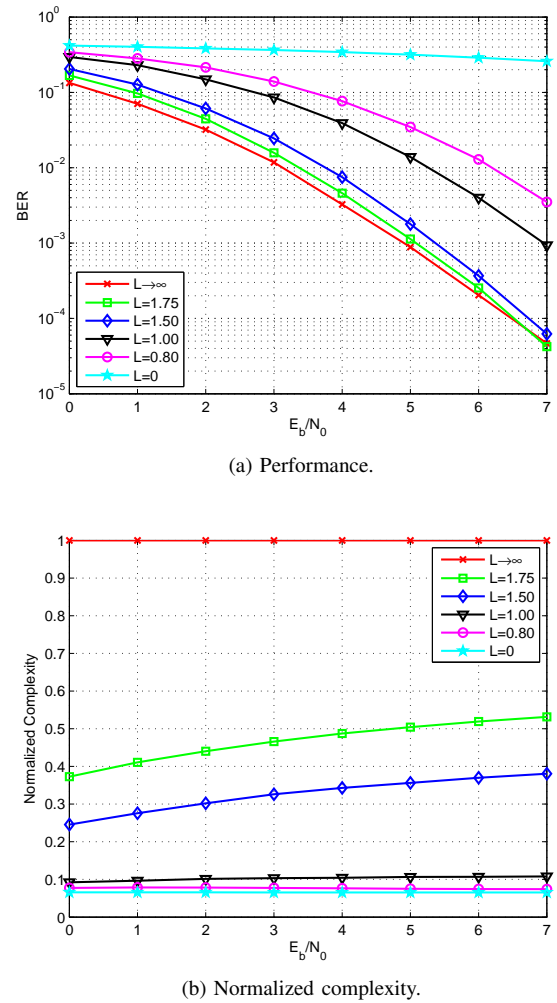


Figure 7: Performance and normalized complexity of the convolutional encoder $G = [54, 64, 74]$.

IV. CONCLUSIONS

A method for reducing the complexity of convolutional decoding based on the samples reliability is proposed. We have concluded that it is possible to achieve a performance close to the Viterbi algorithm, but with much less complexity. The complexity was measured by the number of remaining branches in the trellis, which was validated by the decoding simulation time.

REFERENCES

- [1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.
- [2] Haccoun, D., Caron, M., Nabli, M., "Complexity reduction of the Viterbi algorithm using doubly complementary convolutional codes," *IEEE Conference Proceedings*, pp. 408 - 411, Jan. 1999.
- [3] J.-C. Dany et al., "Low complexity algorithm for the decoding of convolutional codes of any rate," *IEEE Conference Proceedings*, vol.1, pp. 547 - 551, June 2004.
- [4] Frison, Celso Iwata. "Sub-optimal Multiuser Detector based on Reliable Samples", M. Eng. Thesis, Faculty of Electrical and Computer Engineering - Unicamp, Campinas, Brazil, October, 2009, in Portuguese.