# COMPARISON OF STRUCTURES FOR JOINT EQUALIZATION AND DECODING

*David Declercq*

ETIS ENSEA/UCP/CNRS-8051
6, avenue du Ponceau 95014 Cergy-Pontoise (France)
e-mail: declercq@ensea.fr

**ABSTRACT**

In this paper, we compare two structures for combined Equalization/Detection of linear codes over frequency selective channels. The structures come from different families of codes: convolutionnal codes on one hand and parity check block codes on the other hand. First, we show that the joint receiver process corresponds to an iterative belief propagation schedule on graphical representations. Then, we draw and comment the simulation results for various codes and channel choices.

## 1. INTRODUCTION

There has been a increasing interest in coding theory since the introduction of turbo-codes in 1993 [1]. As a first consequence, many new coding and/or decoding techniques have been proposed [2, 3]. It turns out that all good codes are random-like codes and that they share a common decoding algorithm: the Belief Propagation on graphical representations [4]. It has been demonstrated that the best codes representations are *factor graphs* since they are powerful tools to develop decoding algorithms and can be easily generalized to other communication problems. As a matter of fact, one may use belief propagation on graphs for any problem that is composed of a concatenation of two or more blocks, provided that probability density fonctions can be passed between the blocks at the receiver. In this paper, we are interested in joint equalization of frequency selective channels and decoding. The two receiver components are the equalizer and the decoder as described on figure 1. The receiver proceeds iteratively on a block of received data in order to build estimations of the transmitted symbols $u_k$. If convergence is achieved, the receiver output is the optimum solution (maximum *a posteriori*) of the *joint* equalization and decoding problem. In the communication chain of figure 1, we have added a permuter in the transmitter and an inverse permuter in the receiver. The advantage of permuting the information bits is a consequence of the turbo-codes structure and is argued in section 3.

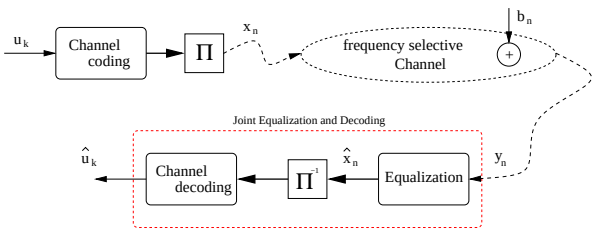


**Fig. 1**. Communication problem that we address.

As long as we can build a graphical representation for the concatenation of the code and the channel, we may apply belief propagation on it. Therefore various coding schemes can be considered and we propose in this paper to compare the performance of combined Equalization-Decoding for two types of codes: *(i)* systematic recursive convolutional codes (SRC codes) and *(ii)* Gallager low density parity check codes (LDPC codes). Which code family is best suited when the channel is frequency selective ? In this contribution, we try to answer this question through extensive simulations. The paper is organized as follow: in the second section, we present the factor graphs of the SRC code, the LDPC code and the frequency selective channel. Together with the graphs, the belief propagation algorithm is pointed out in each case. In section 3, the joint problem of equalization and decoding is addressed and the proposed algorithm is depicted. In section 4, we show the simulation results and discuss the advantages and drawbacks of each structure. Section 5 concludes the paper with some outlooks.

## 2. FACTOR GRAPHS AND BELIEF PROPAGATION

Factor Graphs have been proposed by Wiberg [5] as a generalization of Tanner graphs in coding theory. They are bipartite representations of systems composed of data nodes and function nodes. The data nodes represent observations, input symbols or hidden state variables while the function nodes describe how their adjacent data nodes interact. The branches of the graph carry probability weights that comes in and out the data nodes: they are either interpreted as *a priori* or *a posteriori* information. Belief propagation in a graph depicts how the weights are updated until a fixed point has been reached [4]. It can be shown that exact *a posteriori* weights can be computed if the factor graph is indeed a tree, that is there is no cycles in the graph. Besides, if the cycles in the graph are "sufficiently" long, iterative decoding with probability propagation yields excellent (though approximate) results, close to optimum performance.

### 2.1. Factor Graphs for Convolutional Codes

Throughout this paper, we shall only consider rate $\frac{1}{2}$ systematic recursive convolutional codes. A convolutional code is specified by its state length $\nu$ and its generator polynomials $G_n$ and $G_d$, writen in octal form [6]. All convolutional codes are finite state Markov chains, and then have a very simple graphical representation (cf. figure 2).

Lets $u_n$ be the input bits and $\mathbf{S}_n$ the state vector of the encoder. One output of the encoder is $u_n$ because of the systematic part of the code and the other output $r_n$ is the redundant bit, computed from the previous state $\mathbf{S}_{n-1}$ and the present information bit $u_n$. The messages on the branches that are connected to a state vector are probability weight vectors of size $\nu$ and the function nodes are indicator functions that tell whether there is a connection between

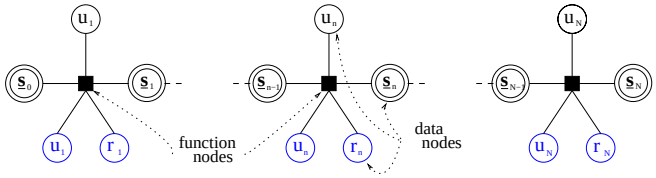


**Fig. 2**. Factor graph of a convolutional code.

the neighboring data nodes. In mathematical form, we have:

$$F(u_n, \mathbf{S}_n, \mathbf{S}_{n-1}, r_n) = \mathbb{1}_{[u_n, \mathbf{S}_n, \mathbf{S}_{n-1}, r_n]}$$

The well known BCJR algorithm [7] which provides the posterior weigths of the information bits is an instance of Belief Propagation in the graph described on figure 2. The propagation schedule that we used is the same than the one proposed for the BCJR algorithm, which is to compute the probability weights for the state nodes $\mathbf{S}_n$ (these weights are called $\alpha_n$ and $\beta_n$ in the BCJR algorithm). Then, the *a posteriori* distributions for each $u_n$ and each $r_n$ are computed. For more details, one can refer to [4]. Note that the graph in figure 2 is actually a tree and belief propagation converges in one iteration to the maximum *a posteriori* (MAP) solution.

### 2.2. Factor Graphs for Gallager Codes

The other family of codes we investigate are Gallager block codes which are defined by a *parity check matrix*. These block codes have been proposed by Gallager in 1963, together with a stochastic decoding algorithm which is very close to belief propagation. MacKay & al. have rediscovered and extended LDPC Gallager codes recently [2] and have shown that Gallager codes can be easily decoded with iterations of belief propagation on their factor graph (cf. figure 3).

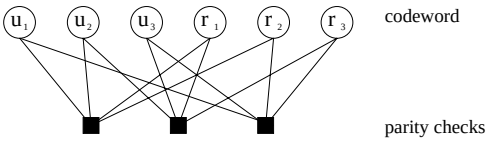


**Fig. 3**. Factor Graph of a LDPC Gallager code $(2, 4)$.

The factor graph of a Gallager code is even simpler than for convolutional codes. Information bits and redundancy bits are linked through function nodes that indicate if the sum of the bits is even. For example, the first function node in figure 3 is simply

$$F(u_1, u_2, r_1, r_2) = \mathbb{1}_{u_1+u_2+r_1+r_2=0 (mod 2)}$$

A Gallager code is regular if each node participates to the same number $t$ of check functions and if each check is connected to the same number $t_r$ of nodes. The Parity check matrix has then constant column weight $t$ and constant row weight $t_r$. If moreover $t$ and $t_r$ are small, the parity matrix is sparse, reason why Gallager codes are named *low density*. Throughout this paper, we will consider rate $R = \frac{t_r - t}{t_r} = \frac{1}{2}$ LDPC codes. Figure 3 gives the example of a Gallager $(t, t_r) = (2, 4)$ code with codeword length $N = 6$. For more details on Gallager codes and clever encoding algorithms, refer to [2]. Despite SRC codes' graphs, LDPC codes' graphs have cycles and many iterations of a belief propagation schedule are needed to reach a fixed state. We have adopted the so-called *flooding schedule* as described in [8] in order to decode our LDPC codes.

### 2.3. Factor Graphs for Frequency Selective Channels

Basically, the factor graph of a frequency selective channel is almost the same as for a SRC code (cf. figure 4). $x_{1\to N}$ are the transmitted symbols, $z_{1\to N}$ are the outputs of the dispersive channel with impulse response $h_{0\to L}$ and $y_{1\to N}$ are noisy versions of $z_{1\to N}$ (observed values). $\mathbf{S}_{x\,0\to N}$ are the state vectors of the channel: $\mathbf{S}_{xn} = [x_{n-1} \dots x_{n-L}]^T$.

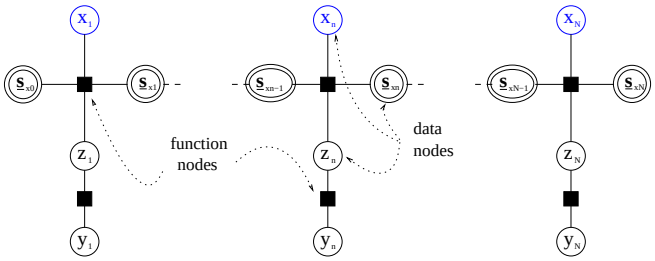


**Fig. 4**. Factor Graph of a frequency selective channel.

In order to make the description of the graph complete, we have to give the expressions for the function nodes. The "top" function nodes in the graph of figure 4 represent the evolution equations of the channel Markov chain. These nodes are therefore indicator functions that give the state value $\mathbf{S}_{xn}$ and the output of the convolutive channel $z_n$, knowing the input of the channel $x_n$ and its previous state $\mathbf{S}_{xn-1}$:

$$F(x_n, \mathbf{S}_{xn}, \mathbf{S}_{xn-1}, z_n) = \mathbb{1}_{[x_n, \mathbf{S}_{xn}, \mathbf{S}_{xn-1}, z_n]}$$

The "bottom" function nodes in figure 4 are nothing else than the Likelihood values of the channel outputs:

$$F(y_n, z_n) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(-\frac{1}{2\sigma_b^2}(y_n - z_n)^2\right)$$

Using belief propagation on the graph of figure 4 is equivalent to using the MAP equalizer proposed by Bahl et al. [7]. Just like the convolutional code, the graph is a tree and belief propagation converges in one iteration.

## 3. JOINT EQUALIZATION AND DETECTION

The idea behind joint processing at the receiver is to pass soft information from one block to another in a clever way. The best strategy is to pass *a posteriori* probability densities coming out one block and use them as *a priori* densities for the next block. The iterative propagation of the densities between two or more receiver components is the base of turbo-decoding and is usually named the *"turbo-principle"*. In order to investigate the turbo-principle on a joint equalizer/decoder, we have to build the factor graph of the concatenation of the encoder and the channel.

### 3.1. SRC codes

In [9] it is argued that a permuter must be added before the transmission in order to separate the state spaces of the encoder and of

the channel. When a permuter is present, the structure of the receiver behaves exactly as a Turbo-decoder: two BCJR algorithms separated by a random permuter. In the early literature about turbo-decoding [1], a soft value called *"extrinsec" information* was computed in order to pass the probability weights from one block to another in an iterative fashion. A simpler and clearer way to interpret this soft information is to build the graph for the concatenation of the encoder, the permuter and the channel and to apply belief propagation on it. The graph is drawn on figure 5 where the two main components are described in details in section 2. This time, the graph has cycles and the permuter allow to turn the short cycles into long ones. If there is no short cycle, the graph is viewed from one particular node *almost* as a tree and belief propagation will perform well.

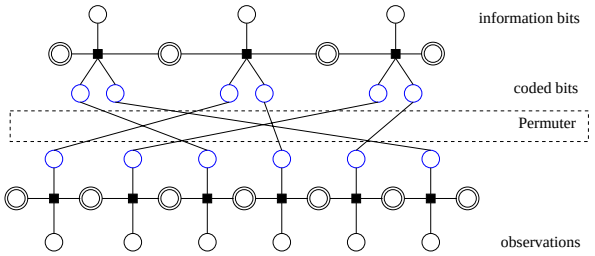


**Fig. 5**. Factor graph for SRC codes and frequency selective channels. The top of the graph represents the encoder and the bottom represents the channel.

The belief propagation schedule is a generalization of the turbo-decoding algorithm: a forward/backward schedule is applied to the channel graph (which is the BCJR equalizer) then the posterior probabilities are permuted and used as prior information for the forward/backward schedule in the SRC code graph. This describes one iteration of turbo-principle on the joint graph of figure 5. Many such iterations must be performed in order to achieve convergence to a fixed point.

### 3.2. LDPC codes

For LDPC codes, there is no need to consider a permuter in the transmitter since the sparseness of the parity check matrix ensures that there is no short cycle in the corresponding graph (provided that the LDPC code has been carefully chosen). This is an advantage over the SRC codes because we can easily close the channel graph (force the last state node to $0$), which is not possible in a simple way for the SRC structure of figure 5. The graph for joint equalization and decoding of LDPC codes is given in figure 6.

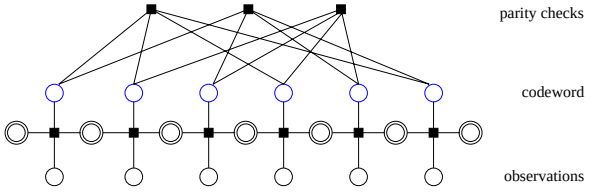


**Fig. 6**. Factor graph for LDPC codes and frequency selective channels. The top of the graph represents the encoder and the bottom represents the channel.

We have adopted the same kind of schedule as in the previous section: one performs iteratively belief propagation on the channel graph then on the code graph. Notice that a few iterations of belief propagation on the code graph have to be done before propagating the probability weights to the channel graph because the LDPC decoder works itself in an iterative fashion.

## 4. SIMULATION RESULTS

In order to make a fair comparison between the two presented structures, we have considered the same assumptions for the transmission. We have chosen to consider a *small* code and a *powerfull* code for each family of codes and two frequency selective channels which are known as *hard to equalize*, namely the Proakis-B and Proakis-C channels [6]. The two SRC codes are rate $\frac{1}{2}$: $(1, 5/7)$ is the *small* code of constraint length $\nu = 2$ and $(1, 171/133)$ is the *powerfull* code of constraint length $\nu = 7$. The two LDPC codes are also rate $\frac{1}{2}$ codes and are regular Gallager codes. LDPC$(2, 4)$ is the *small* code and LDPC$(3, 6)$ is the *powerfull* code. The transmission is BPSK modulated and perfect knowledge of the channel taps is assumed. The size of the transmitted blocks has been chosen small $N = 1000$, so that it could correspond to a realistic transmission. The size of the permuter is then $N = 1000$ and the length of a Gallager codeword is also $N = 1000$. It was not possible to draw all the simulation results, and we have chosen to present only the case of the *small* code and Proakis-C channel, and of the *powerfull* code and Proakis-B channel for each family of code. The Simulations have been done with $10^7$ bits and are reported in figures 7-10. For each figure, we have drawn the performance of the code over an AWGN channel, which corresponds to the lower bound of the transmission. We have also drawn the iterations number 1, 2, 5 and 15 of the turbo-receiver.

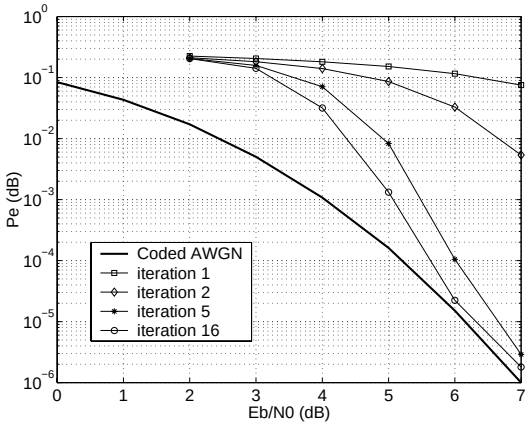


**Fig. 7**. SRC$(1, 5/7)$ code and Proakis-C channel

As a first reamark, we can notice that joint equalization and detection performs really well since the iterated process is very close to the optimum curve. Another common remark is that the performance are closer to the optimum as the signal to noise ratio increases. This could certainly be explained with the same kind of analysis as in [10] and we will investigate this aspect in future work. Regarding the comparison between SRC and LDPC codes for joint equalization and detection, it appears that SRC codes allow the joint iterative receiver to converge to its optimum performance at high SNR. This is not the case for LDPC codes since there is a gap between the result of belief propagation and the coded AWGN lower bound: almost $1dB$ at $Pe = 10^{-5}$. The reason could be twofold: it could come from the very structure

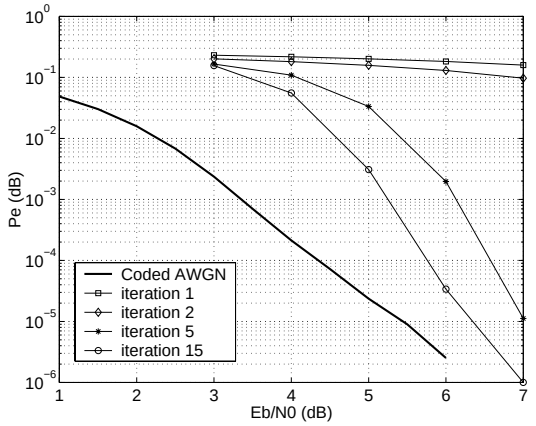


**Fig. 8**. LDPC$(2, 4)$ code and Proakis-C channel

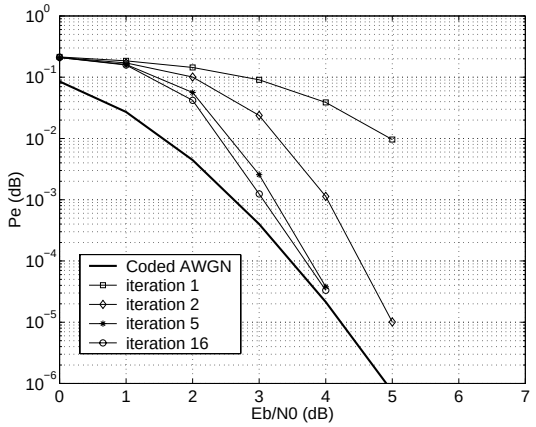


**Fig. 9**. SRC$(1, 171/133)$ code and Proakis-B channel

of LDPC codes or from the fact that this code must be decoded iteratively. We have no clear answer to venture. Despite this drawback, LDPC codes achieve the same overall performance than SRC codes for the same decoding complexity. Indeed, for the Proakis-B channel at $Eb/N_0 = 7dB$, both codes provide an error probability of $\simeq 10^{-6}$ and for Proakis-C channel at $Eb/N_0 = 5dB$ we obtain $P_e \simeq 10^{-5}$ (recall that the codes for the two channels are differents).

## 5. CONCLUSION AND OUTLOOKS

As a conclusion, this comparison study has shown that there is no clear advantage between SRC codes and block LDPC codes for joint equalization and decoding. There is a tight advantage for SRC codes since the iterative receiver converges to its optimum performance, though there is a gap of almost $1dB$ for LDPC codes. However, it is a lot easier to build LDPC codes for a wide range of code rates, which is not the case of SRC codes. LDPC codes could therefore be prefered for some applications since they perform as well as SRC codes and they are more versatile. It should be interesting to compare LDPC codes with another family of iteratively decodable codes: namely turbo-codes. This could give some insights about the gap between the iterative receiver performance and the optimum AWGN curve.

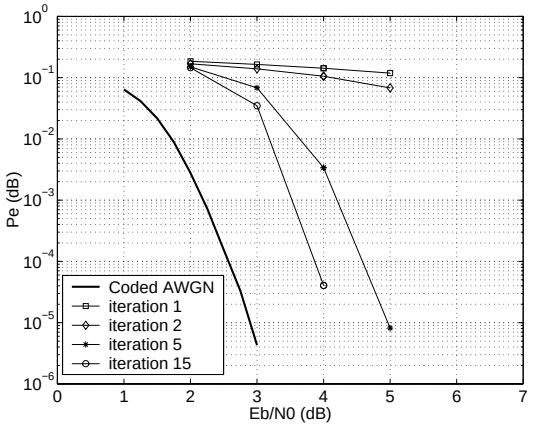


**Fig. 10**. LDPC$(3, 6)$ code and Proakis-B channel

## 6. ACKNOWLEGEMENTS

I would like to thank Leopold Weinberg from university of Lille (France) who helped me to fully understand Gallager Codes. I appreciated specially his kindness in helping me go through the computer simulations.

## 7. REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimasjshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes.," in *IEEE Int. Conf. Comm. (ICC)*, Geneva (Switzerland), 1993, pp. 1064–1070.

[2] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theo.*, Jan 1999.

[3] D. Divsalar, H. Jin, and R.J. McEliece, "Coding theorems for 'turbo-like' codes," in *Allerton conf. on Communication, Control and Computing.*, Allerton (Il, USA), 1998, pp. 201–210.

[4] F. R. Kschischang, B. J. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm.," *submitted to IEEE Trans. Inf. Theo.*, July 1998.

[5] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. thesis, Linköping studies in science and technology (Sweden), 1996.

[6] J.G. Proakis, *Digital communications ($3^{rd}$ ed.)*, Mc Graw-Hill, New York, 1995.

[7] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rategood error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theo.*, pp. 284–287, March 1974.

[8] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models.," *IEEE J. Selec. Areas in Comm.*, vol. 16, no. 1, Jan. 1998.

[9] C. Douillard et al., "Iterative correction of intersymbol interference: Turbo-equalization," *Euro. Trans. on Telecom.*, vol. 6, pp. 507–511, 1995.

[10] D. Divsalar, S. Dolinar, and F. Pollara, "Low complexity turbo-like codes," in *IEEE symp. on Turbo-Codes*, Brest (France), 2000, pp. 73–80.