Joint source-channel turbo decoding of entropy-coded sources

Arnaud Guyader, Eric Fabre,
Christine Guillemot, Matthias Robert*

Abstract

We analyze the dependencies between the variables involved in the source and channel coding chain. This analysis is carried out in the framework of Bayesian networks, which provide both an intuitive representation for the global model of the coding chain, and a way of deriving joint (soft) decoding algorithms. Three sources of dependencies are involved in the chain : 1/ the source model, a Markov chain of symbols, 2/ the source coder model, based on a variable length code (VLC), for example a Huffman code, and 3/ the channel coder, based on a convolutional error correcting code. Joint decoding relying on the hidden Markov model (HMM) of the global coding chain is intractable, except in trivial cases. We advocate instead an iterative procedure inspired from serial turbo codes, in which the three models of the coding chain are used alternately. This idea of using separately each factor of a big product model inside an iterative procedure usually requires the presence of an interleaver between successive components. We show that only one interleaver is necessary here, placed between the source coder and the channel coder. The decoding scheme we propose can be viewed as a turbo algorithm using alternately the intersymbol correlation due to the Markov source and the redundancy introduced by the channel code. The intermediary element, the source coder model, is used as a translator of soft information from the bit clock to the symbol clock.

keywords : Joint source-channel decoding, probabilistic inference, Bayesian network, iterative decoding, soft decoding, turbo-code, entropy coding, data compression, variable length code.

1 Introduction

The advent of wireless communications, often characterized by narrowband and noisy channels, is creating challenging problems in the area of coding. Design principles prevailing so far and stemming from Shannon's source and channel separation theorem are being re-considered. The separation theorem, stating that source and channel optimum performance bounds can be approached as closely as desired by designing independently source and channel coding strategies, holds only under asymptotic conditions where both codes are allowed infinite length and complexity. If the design of the system is heavily constrained in terms of complexity or delay, source and channel coders can be largely suboptimal.

Joint source and channel coding and decoding have therefore gained considerable attention as viable alternatives for reliable communication across noisy channels. For joint coding, the

*All authors are with IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex, FRANCE. E-mail :firstname.lastname@irisa.fr, Fax:+33299842531, Phone:+33299847429

idea relies often on capitalizing on source coder (SC) suboptimality, by exploiting residual source redundancy (the so-called "excess-rate"). As a consequence, joint source-channel decoding must make use of both forms of dependencies. First attempts at joint source-channel decoding considered fixed rate source coding systems [1, 2, 3, 4]. However, the wide use of variable length codes (VLCs) in data compression has motivated recent consideration of variable length coded streams, focusing first on robust decoding of such bit streams. In [5, 6, 7], a Markov source (MS) of symbols is assumed, which feeds a VLC source coder (Huffman coder). A major weakness of VLC coded streams comes from the lack of synchronization between the symbol clock and the bit clock, which makes them very sensitive to channel noise. A joint VLC decoding relying on the residual redundancy of the MS has been shown to reduce this effect. It is only lately that models incorporating both VLC-encoded sources and channel codes (CC) have been considered [8, 9, 10, 11, 12]. The authors in [8] derive a global stochastic automaton model of the transmitted bit stream by computing the product of the separate models for the Markov source (MS), the source coder (SC) and the channel coder (CC). The resulting automaton is used to perform a MAP decoding with the Viterbi algorithm. The approach provides the optimal joint decoding of the chain, but remains intractable for realistic applications because of state-complexity explosion phenomenon. In [10, 11, 12, 13], the authors remove the memory assumption for the source. They consider a general variable length SC followed by a convolutional CC, these two components being separated by an interleaver. They propose a turbo-like iterative decoder for estimating the transmitted symbol stream, which alternates channel decoding and VLC decoding. This solution has the advantage of using one model at a time, thus avoiding the state explosion phenomenon.

The purpose of this paper is to extend this turbo approach to a general coding chain, encompassing as particular cases the models of the papers above. The chain is composed of a Markov source of symbols, followed by a variable length source coder transforming symbols into information bits, the latter feeding a convolutional channel coder[1]. We also assume that both the number of transmitted symbols and the corresponding number of bits in the coded sequence are known. The former is usually determined a priori by the transmission protocol, while the latter can be determined at the receiver by isolating a prefix and a postfix of the bistream. Such an assumption does not reduce the generality of our framework : the difficulty is in the treatment of this information. Estimation algorithms become simpler when this information is not known.

We focus on an analysis and modeling of the dependencies between the variables involved in the complete chain of source and channel coding, by means of the Bayesian network formalism. Bayesian networks are a natural tool to analyze the structure both of stochastic dependencies and of constraints between variables, through a graphical representation. They are also the relevant way of reading out conditional independence relations in a model, which form the basis of fast estimation algorithms (e.g. the Kalman filter, the BCJR, the Viterbi algorithm, etc.). Indeed, the structure of Bayesian inference algorithms, either exact or approximate, and for several criteria, can often been derived "automatically" from the graph. We therefore address the problem of joint source and channel decoding in this framework.

As in the early work of [8], our starting point is a state space model of the three different elements in the chain : the symbol source, the source coder and the channel coder. These models are cascaded to produce the bitstream sent over the channel, and the randomness of variables is introduced by assuming a white noise input of the cascade. The product of these three automata induce immediately a state variable model of the bitstream : the triple of states – one state

[1]The present work extends directly to semi-Markov models for the source. Also, we consider a convolutional channel coder, but bloc codes induce the same type of difficulties.

for each model – appears to be a Markov chain, the transitions of which generate the sequence of output bits, that are sent over the channel. The observed output of a memoryless channel corresponds to noisy measurements of these bits. Recovering the transmitted sequence of source symbols from these noisy measurements is equivalent to inferring the sequence of model states. Therefore we are exactly in the HMM framework, for which fast estimation algorithms are well known.

This nice picture suffers from two difficulties, however. First of all, the presence of *two time indexes*: the symbol clock, and the bit clock. The input of the source model is an i.i.d. sequence that produces symbols with the right joint distribution (we will assume a Markov source in the sequel). Input and output sequences are synchronous and indexed by the symbol clock. At the other extremity, the channel coder gets a (correlated) sequence of useful bits, to which some redundancy is incorporated. Input and output time indexes are proportional, the coefficient being the rate of the error correcting code. No difficulty here, and we define the bit clock as the index of the channel coder input. By contrast, the central element, i.e. the source coder, receives a sequence of (correlated) source symbols, and outputs variable length codewords. So it operates a clock conversion with a varying rate. Actually, for a given number of source symbols, the number of bits of the coded sequence is a random variable, which is quite unusual. The second difficulty is more classical: it comes from the fact that *the state space dimension of the product model explodes* in most practical cases, so that a direct application of usual techniques is unaffordable, except in trivial cases. In this paper, we thus rely on properties evidenced by serial turbo-codes to design an estimation strategy: instead of using the big product model, inference can be done in an iterative way, making use of part of the global model at each time.

In details, this takes the following form. Instead of building the Markov chain of the product model, one can directly consider the Bayesian network corresponding to the serial connection of the three HMMs, one for the source, one for the source coder and one for the channel coder. This simplifies the Bayesian network since "smaller" variables are involved. However, beyond the time index difficulty, this results in a complex Bayesian network with a high number of short cycles, and thus not amenable to fast estimation algorithms. It was observed with turbo-codes [14, 15, 16], that efficient approximate estimators can be obtained by running a belief propagation algorithm on a *cyclic* Bayesian network (which is theoretically "illegal"), provided the cycles are long enough. The great innovation of turbo-codes is that the simple introduction of an interleaver between two models can make short cycles become long. Adopting this principle, one can design an iterative estimator working alternately on each factor of the product model, with significant gain in complexity.

We use this idea in the following way, focusing first on a constant length source code (CLC), in order to separate difficulties. As it was already suggested in [10], we introduce an interleaver between the source coder and the channel coder. The Bayesian Network formalism shows that there is no need of a second interleaver separating the Markov source and the source coder. This allows the construction of an iterative soft decoder alternating between the CC model, and the joint model of the MS + SC[2], with the bit clock as time index. But the idea can be pushed further. The joint MS+SC model can actually be processed optimally by a sequential use of the SC model, followed by the MS model. We end up with an iterative procedure between the two sources of redundancy (the MS and the CC), where the intermediary SC model is used as a translator of soft information from the bit clock to the symbol clock.

When we move to variable length source codes, a new phenomenon comes into the picture:

[2]By contrast, [10] is assuming an i.i.d. source, which makes the source model useless.

for a fixed number of symbols, the number of output bits is random, which makes the structure of the Bayesian network random. This difficulty remains even if both the number of symbols and the number of bits are known, since the segmentation of the bitstream into codewords remains random. But surprisingly, all algorithms developed in the CLC case extend to VLCs, which is a new result. In particular, even in the case of VLCs, there is no need of an extra interleaver separating the MS and the SC : a successive use of these models is optimal for joint decoding of the pair.

The rest of the paper is organized as follows. Section 2 describes part of the notation we use. Section 3 revisits briefly classical estimation algorithms to give them a graphical interpretation, on which we rely in the sequel. Section 4 addresses modelling issues in the case of constant length source codes (CLC), in order to focus on the structure of the iterative algorithm based on three models. However, we distinguish the two time indexes. Section 5 relies on this material to study and solve the extra difficulty introduced by variable length codes (VLC). This appears to be only a technical extension, that doesn't change the ideas of section 4, but only makes them less obviously applicable. Finally, experimental results are described in section 6, in which we observe the resynchronization properties of the algorithm.

2 Notations

Let $S = S_1 \dots S_K$ be the sequence of source symbols, coded into a sequence of information bits $U = U_1 \dots U_N$, by means of a variable length source code (e.g. a Huffman code). k and n represent generic time indexes for the symbol clock and the bit clock, respectively. We denote by $\bar{U}_k$ the codeword corresponding to S_k, so $\bar{U} = \bar{U}_1 \dots \bar{U}_K$ represents bitstream U segmented into codewords. As mentioned in the introduction, both K and N are assumed to be known, since the difficulty is in the treatment of this information (estimation algorithms become simpler when one of these lengths is unknown). Observe that the length N of the information bit stream is a random variable, function of S. A sequence of redundant bits $R = R_1 \dots R_M$ is constructed from U by means of a (possibly punctured) systematic convolutional error correcting code. In the triple (S, U, R), all the randomness comes from S, since U and R are deterministic functions of S. The bitstream (U, R) is sent over a memoryless channel and received as measurements (Y, Z) (see fig. 1) ; so the problem we address consists in estimating S given the observed values y and z. We reserve capital letters to random variables, and small letters to values of these variables. For handling ranges of variables, we use the notation $X_u^v = \{X_u, X_{u+1}, \dots, X_v\}$ or X_I where I is the index set $\{u, u+1, \dots, v\}$. We omit I when all indexes are taken. Other notations are defined later in the body of the paper.

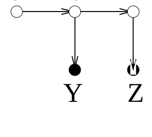


Figure 1: *A graphical model depicting dependencies of processes S, U, R and measurements Y and Z at the output of the channel. This graph states that $\mathbb{P}(S, U, R, Y, Z)$ factorizes according to $\mathbb{P}(S) \cdot \mathbb{P}(U|S) \cdot \mathbb{P}(R|U) \cdot \mathbb{P}(Y|U) \cdot \mathbb{P}(Z|R)$, where $\mathbb{P}(U|S)$ and $\mathbb{P}(R|U)$ are singular, i.e. describe constraints. $\mathbb{P}(Y|U)$ and $\mathbb{P}(Z|R)$ are defined by channel noise. White dots represent unobserved variables to estimate, and black dots stand for observed variables.*

3 Background on algorithms

In the sequel, we analyze the structure of the coding chain in the framework of Bayesian networks [17]. Bayesian networks (BN) are the most natural tool to display the structure both of stochastic dependencies and of constraints between random variables. They also provide a powerful way of reading out conditional independence relations in a model, and it is well known that such relations are the key to fast estimation algorithms. Indeed, the structure of Bayesian inference algorithms, either exact or approximate, and for several criteria, can been derived "automatically" from the graphical representation of the process. In particular, very efficient algorithms exist *as far as this graph is a tree*, where efficiency refers to a linear complexity with respect to the number of variables. Such algorithms are obtained by combining simple primitive operations : *propagation, update* and *merge* (see [18] or section 2 of [19]). There exist numerous ways of combining these primitives, which makes the classification of estimation strategies quite difficult, except with respect to the estimation criterion they use. Moreover, many communities (re-)discovered independently some of these strategies, whence a large variety of names for very similar algorithms (e.g. Kalman smoother, Raugh-Tung-Striebel algorithm, BCJR algorithm, belief propagation, sum-product algorithm, etc.).

In this section, we briefly review some of these strategies for a standard Markov process, which is enough for the sequel. The classification follows 1/ the estimation criterion, and 2/ the general organization of computations (either organized in sweeps, or "graph blind"). Our purpose is to gather here the equational part of algorithms, in order to rely later on graphical arguments to explain the organization of computations for more complex structures.

The Markov process to estimate is $X = X_1 \dots X_N$. The factorization $\mathbb{P}(X) = \prod_n \mathbb{P}(X_n|X_{n-1})$ is graphically represented by the oriented chain of fig. 2. The process X is "hidden," i.e. one only gets information about X through the (noisy) measurement process $Y = Y_1 \dots Y_N$. Measurements are assumed to be conditionally independent given X, i.e. $\mathbb{P}(Y|X) = \prod_n \mathbb{P}(Y_n|X)$. They are also assumed to be local, where locality means either that Y_n measures X_n only, i.e. $\mathbb{P}(Y_n|X) = \mathbb{P}(Y_n|X_n)$, or that Y_n measures the transition from X_{n-1} to X_n, i.e. $\mathbb{P}(Y_n|X) = \mathbb{P}(Y_n|X_{n-1}, X_n)$.

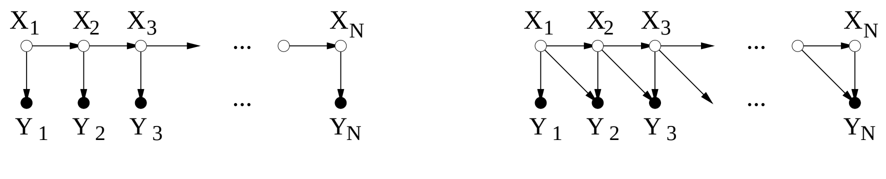

X_1 X_2 X_3 ... X_N
Y_1 Y_2 Y_3 ... Y_N
X_1 X_2 X_3 ... X_N
Y_1 Y_2 Y_3 ... Y_N

Figure 2: *A Markov process X and a measurement process Y, on variables of X (left-hand side) or on transitions of X (right-hand side).*

3.1 MPM estimates

3.1.1 "Organized" strategies

MPM stands for[3] "maximum of posterior marginals," which means that each X_n is estimated individually according to

$$\hat{X}_n = \arg\max_{x_n} \mathbb{P}(X_n = x_n|Y) \tag{1}$$

[3]In the image processing community, MPM is also read as "marginal posterior modes."

Hence the objective is to obtain these posterior marginals. Computations can be organized around the factorization

$$\mathbb{P}(X_n|Y) \quad \propto \quad \mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(Y_{n+1}^N|X_n) \tag{2}$$

where $\propto$ denotes an obvious renormalization. The Markov property allows a recursive computation of both terms of the right-hand side. The "forward" sweep concerns the first term

$$\mathbb{P}(X_n|Y_1^n) \quad \propto \quad P(X_n|Y_1^{n-1}) \cdot \mathbb{P}(Y_n|X_n) \tag{3}$$

$$\text{where } \mathbb{P}(X_n|Y_1^{n-1}) \quad = \quad \sum_{x_{n-1}} \mathbb{P}(X_n|X_{n-1} = x_{n-1}) \cdot \mathbb{P}(X_{n-1} = x_{n-1}|Y_1^{n-1}) \tag{4}$$

Equation (4) is usually referred to as the *propagation* or *prediction* step. In the *update* step (3), $\mathbb{P}(Y_n|X_n)$ assumes Y_n measures X_n ; it must be replaced by $\mathbb{P}(Y_n|X_{n-1}, X_n)$ if Y_n measures the transition from X_{n-1} to X_n (from now on, we omit mentioning this detail). The "backward" sweep provides the second term in (2)

$$\mathbb{P}(Y_{n+1}^N|X_n) \quad \propto \quad \sum_{x_{n+1}} \mathbb{P}(X_{n+1} = x_{n+1}|X_n) \cdot \mathbb{P}(Y_{n+2}^N|X_{n+1} = x_{n+1}) \cdot \mathbb{P}(Y_{n+1}|X_{n+1} = x_{n+1}) \tag{5}$$

Since this quantity goes to zero as the number of measurements augments, it is often handled in a renormalized form (over variable X_n), whence the $\propto$ in (5), which has no influence on (2). This two-sweep organization of computations is sometimes called the BCJR algorithm [20] and is well adapted to a definition of $\mathbb{P}(X)$ through transition probabilities $\mathbb{P}(X_n|X_{n-1})$. A more symmetric version drops the left-right orientation in the factorization of $\mathbb{P}(X)$ by relying on

$$\mathbb{P}(X_n|Y) \quad \propto \quad \frac{\mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(X_n|Y_{n+1}^N)}{\mathbb{P}(X_n)} \tag{6}$$

which is a *merge* of two lateral conditional distributions on X_n. The second term of the numerator can be recursively obtained by (4,3) with a backward factorization of $\mathbb{P}(X)$. So (6) requires $\mathbb{P}(X_n|X_{n+1})$ and $\mathbb{P}(X_n)$.

Remark : the posterior marginal $\mathbb{P}(X_n, X_{n+1}|Y)$ derives immediately from byproducts of an MPM estimation algorithm. For example, if factorization (2) has been chosen to base computations, one has

$$\mathbb{P}(X_n, X_{n+1}|Y) \quad \propto \quad \mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(X_{n+1}|X_n) \cdot \mathbb{P}(Y_{n+1}|X_{n+1}) \cdot \mathbb{P}(Y_{n+1}^N|X_{n+1}) \tag{7}$$

In other words, an MPM estimation algorithm also provides the posterior distribution on *transitions* of X.

3.1.2 Graph-blind strategies

The above estimation strategies organize computations into two sweeps or recursions, thus following closely the graph of figure 2. By contrast, other message passing algorithms only specify local computations and leave unspecified the general organization of message circulations on the graph. Such methods appear in the decoding of sparse parity check codes, for example. Because no global knowledge of the graph is necessary, we refer to this family as "graph-blind methods."

The idea is to reach $\mathbb{P}(X_n|Y)$ by increasing the number of measurements in the conditioning part of $\mathbb{P}(X_n|Y_I)$, where $I \subset \{1,2,\dots,N\}$. For our Markov process, let us define variable index sets $L(n) \subset \{1,\dots,n-1\}$ and $R(n) \subset \{n+1,\dots,N\}$. Since $Y_{L(n)}, Y_n$ and $Y_{R(n)}$ are conditionally independent given X_n, the following merge equation holds

$$\mathbb{P}(X_n|Y_{L(n)}, Y_n, Y_{R(n)}) \quad \propto \quad \frac{\mathbb{P}(X_n|Y_{L(n)}) \cdot \mathbb{P}(X_n|Y_n) \cdot \mathbb{P}(X_n|Y_{R(n)})}{\mathbb{P}(X_n)^2} \tag{8}$$

For every X_n, the algorithm maintains incoming messages $\mathbb{P}(X_n|Y_I)$ on every edge around X_n, where Y_I are measurements located beyond that edge from the standpoint of X_n. For instance, the edge (X_{n-1}, X_n) brings $\mathbb{P}(X_n|Y_{L(n)})$ to X_n. These messages are updated by merge and propagation : the message $\mathbb{P}(X_{n-1}|Y_{R(n-1)})$ sent by X_n to X_{n-1} is updated by

$$R(n-1) \quad := \quad R(n) \cup \{n\} \tag{9}$$

$$\mathbb{P}(X_{n-1}|Y_{R(n-1)}) \quad \propto \quad \sum_{x_n} \mathbb{P}(X_{n-1}|X_n = x_n) \cdot \frac{\mathbb{P}(X_n = x_n|Y_n) \cdot \mathbb{P}(X_n = x_n|Y_{R(n)})}{\mathbb{P}(X_n = x_n)} \tag{10}$$

and symmetrically for the message sent to X_{n+1}. Equation (10) can be generalized by leaving the inclusion of Y_n optional. By monotonicity, the only stable state of the algorithm is obtained for $L(n) = \{1,\dots,n-1\}$ and $R(n) = \{n+1,\dots,N\}$ for all n. So, whatever the ordering of updates, (8) finally gives the desired posterior marginals. Notice that variations of the algorithm can be obtained by defining messages and updates on other factorizations than (8), for example

$$\mathbb{P}(X_n|Y_{L(n)}, Y_n, Y_{R(n)}) \quad \propto \quad \mathbb{P}(Y_{L(n)}|X_n) \cdot \mathbb{P}(Y_n|X_n) \cdot \mathbb{P}(Y_{R(n)}|X_n) \cdot \mathbb{P}(X_n) \tag{11}$$

or any other Bayesian equation[4]. Notice also that the organized algorithms presented above come out as particular orderings of updates operations for particular factorizations of $\mathbb{P}(X_n|Y)$.

One specific ordering of computations is worth mentioning here, in the case where Y measures variables of X (not transitions). It starts by computing local posterior distributions $\mathbb{P}(X_n|Y_n)$ for every X_n, and then organizes updates in two sweeps, from left to right and right to left. This strategy corresponds to computing first $\mathbb{P}(X|Y)$ assuming X is a white noise with distribution $\mathbb{P}(X) = \prod_n \mathbb{P}(X_n)$, and then taking correlations into account. We will refer to it as algorithm $\mathcal{A}$ in the sequel.

3.2 MAP estimates

The MAP criterion (maximum *a posteriori*) corresponds to the optimal Bayesian estimation of the whole process X based on all available measurements Y :

$$\hat{X} \quad = \quad \arg\max_x \mathbb{P}(X = x|Y) \tag{12}$$

hence the optimization is over all possible *sequences* x. For $I \subset \{1,\dots,N\}$ an index set and $J = I \setminus \{n\}$, let us define notation $\bar{\mathbb{P}}$ by

$$\bar{\mathbb{P}}(X_n|Y_I) \quad \propto \quad \max_{x_J} \mathbb{P}(X_n, X_J = x_J|Y_I) \tag{13}$$

$$\bar{\mathbb{P}}(Y_I|X_n) \quad \propto \quad \max_{x_J} \mathbb{P}(Y_I, X_J = x_J|X_n) \tag{14}$$

[4]Remark : when Y measures transitions of process X, the index n of measurement Y_n must be counted by $L(n)$.

This definition drops a multiplicative constant, which is useless for the estimation and leaves space to renormalizations (over X_n), which favors the algorithms stability. The optimal sequence $\hat{X}$ can be obtained by gathering local estimates $\hat{X}_n$ defined by

$$\hat{X}_n \quad = \quad \arg\max_{x_n} \bar{\mathbb{P}}(X_n = x_n|Y) \tag{15}$$

hence once again the objective is to obtain these "posterior marginals." Bayes factorizations like (2,6,8) or (11) remain valid for $\bar{\mathbb{P}}$. For example (2) becomes

$$\bar{\mathbb{P}}(X_n|Y) \quad \propto \quad \bar{\mathbb{P}}(X_n|Y_1^n) \cdot \bar{\mathbb{P}}(Y_{n+1}^N|X_n) \tag{16}$$

In the same way, it is straightforward to check that all computations defined on $\mathbb{P}$ extend to $\bar{\mathbb{P}}$ provided the $\sum$ operator is replaced by the *max* operator. Hence all estimation strategies designed for the MPM criterion extend directly to the MAP, which was already mentioned by several authors [21, 22, 23]. For example, the "sum-product" algorithm for the MPM becomes the so-called "max-sum" algorithm for the MAP [23] if one uses the logarithm of $\bar{\mathbb{P}}$, which is usually preferred.

This theoretical result is very useful for designing estimation strategies, but is somehow suboptimal. The final *max* in (12) and some update messages can be avoided by capitalizing on intermediary *max* operations. For example while updating the message $\bar{\mathbb{P}}(X_{n-1}|Y_{R(n-1)})$ in (10), the argument of the *max* over x_n can be stored as a function of X_{n-1} : $X_n^*(X_{n-1})$. When $\hat{X}_{n-1}$ is known, one directly has $\hat{X}_n = X_n^*(\hat{X}_{n-1})$. This trick is well known and implemented in the Viterbi algorithm.

3.3 Extrinsic information

This notion represents an intermediary product of a belief propagation algorithm. It has little relevance by itself but is commonly used for explaining the structure of iterative algorithms. Let X be some random variable to be estimated using two measurements Y and Z. On has the decomposition

$$\mathbb{P}(X|Y,Z) \quad = \quad \mathbb{P}(X) \cdot \frac{\mathbb{P}(Y|X)}{P(Y)} \cdot \frac{\mathbb{P}(Z|X,Y)}{P(Z|Y)} \tag{17}$$

The first term is the *a priori* information about X, the second one the information of the first measurement Y about X, and the last one the remaining information carried by Z about X once Y is known, i.e. the *extrinsic information* $Ext_X(Z|Y)$. It is often determined by

$$Ext_X(Z|Y) \quad = \quad \frac{\mathbb{P}(X|Y,Z)}{\mathbb{P}(X|Y)} \tag{18}$$

Notice that $Ext_X(Z|Y)$ can be handled as the conditional distribution $\mathbb{P}(\Delta|X)$ of a *pseudo* measurement Δ on X. It plays the part of the rightmost term in (2), and hence can be read as a message sent back to X to update an estimate. We use this interpretation in the sequel.

In the context of Markov models, the extrinsic information on X_n is often defined as

$$Ext_{X_n}(Y|Y_n) \quad = \quad Ext_{X_n}(Y_1^{n-1}, Y_{n+1}^N|Y_n) \tag{19}$$

In words, it represents the extra information carried by Y about X_n once the local measurement Y_n is known. The notion of extrinsic information as defined by (18) extends to $\bar{\mathbb{P}}$.

4 Joint source-channel decoding for constant length source codes

To clarify the structure of the algorithms developed for VLC encoded sources, we first consider codewords of constant length l. Constant length codes (CLC) make the problem much simpler: with a VLC, one must both recover symbol boundaries, and transmitted symbols in the noisy bit stream, whereas here the segmentation is known. Although the bit clock is a multiple of the symbol clock, we distinguish them and build models for both time indexes, in order to prepare for the VLC case. We start with a decoder for the pair MS+SC, and prove that there exists an optimal joint decoder using first the SC model, and then the MS model. We then consider the complete chain MS+SC+CC, for which this strategy fails, and propose instead an iterative procedure.

4.1 Decoding of the pair Markov source + source coder

4.1.1 Symbol clock model

For simplicity, we consider a symbol source S described by an order-one Markov process[5]. Symbols S_k are translated into codewords $\bar{U}_k$ by a deterministic function. Thanks to the constant length property, one has $\bar{U}_k = U_{(k-1)l+1}^{kl}$. Gathering measurements $Y_{(k-1)l+1}^{kl}$ into $\bar{Y}_k$, we have a symbol clock model for MS+SC that fits exactly section 3 (fig. 3). Thus estimation algorithms are readily available, with complexity[6] $\mathbf{C}_1 = O(K \cdot |\mathcal{S}|^2)$, where $\mathcal{S}$ is the set of possible source symbols.

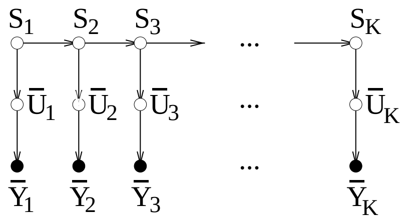


Figure 3: *Bayesian network of a symbol clock model for the pair MS+SC, in the case of a constant length source code. Black dots show the observed variables.*

4.1.2 Bit clock model

Notice that estimating S is equivalent to estimating U. To design a bit clock model for the pair $MS + SC$, we must focus on U and analyze the structure of its distribution. $\mathbb{P}(U)$ can actually be modelled as a semi-Markov process. We have $\mathbb{P}(U) = \mathbb{P}(\bar{U}_1) \cdot \mathbb{P}(\bar{U}_2|\bar{U}_1) \cdots \mathbb{P}(\bar{U}_K|\bar{U}_{K-1})$, so the problem amounts to factorizing further each element.

For the first term, Bayes formula $\mathbb{P}(\bar{U}_1) = \mathbb{P}(U_1) \cdot \mathbb{P}(U_2|U_1) \cdots \mathbb{P}(U_l|U_1^{l-1})$ suggests a stochastic automaton in form of a decision tree for generating codewords $\bar{U}_1$ (fig. 4): each vertex ν of the tree corresponds to a tuple U_1^{i-1}, from which two transitions are possible, one for $U_i = 0$ and one for $U_i = 1$. A bit clock model for $\mathbb{P}(\bar{U}_1)$ follows immediately: let us define X_i as the state (i.e. vertex ν) reached after i transitions of this automaton. Then X_0^l is a Markov process, and U_i is a deterministic function of its transitions, that is of (X_{i-1}, X_i).

[5]The results in this paper extend directly to any semi-Markov source, which allows a longer memory for S.
[6]Complexities are evaluated as the number of multiplications.

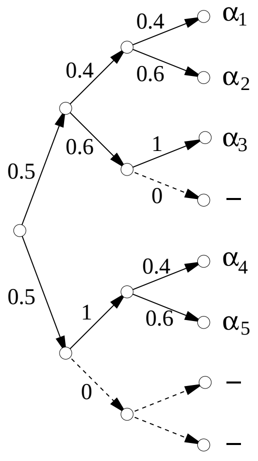


Figure 4: *Example of a stochastic automaton associated to Bayes decomposition of* $\mathbb{P}(\bar{U}_1) = \prod_{i=1}^{l} \mathbb{P}(U_i|U_1^{i-1})$*, for* $l = 3$ *and 5 possible source symbols, with probabilities* (0.08, 0.12, 0.3, 0.2, 0.3)*. Transitions probabilities are mentioned close to the edges. Transitions upward produce* $U_i = 1$ *while transitions downward produce* $U_i = 0$*.*

To factorize the remaining terms $\mathbb{P}(\bar{U}_k|\bar{U}_{k-1})$ according to the bit clock, we must proceed in the same way, but *for every possible value of* $\bar{U}_{k-1}$, or equivalently of S_{k-1}. In other words, for $k \geq 2$, we must keep track of the last symbol produced. Let us define the general state variable X_n as a pair (σ, ν) where σ is the value of the last completed symbol S_k, with $k = \lfloor n/l \rfloor$, and ν is the current state of the stochastic automaton describing the construction of the next symbol, following $\mathbb{P}(S_{k+1}|S_k)$. Once again X is a Markov process, the transitions of which produce U (fig. 5).

The last step of the construction consists in connecting the local bit clock models for $\mathbb{P}(S_{k+1}|S_k)$. This amounts to identifying each terminal state X_{kl} with the initial state of the next symbol. There are two practical ways to do so. Solution A views each leafnode ν as a root of the next tree ; this preserves the leaves of the tree as possible values for ν, and removes the rootnode. Solution B is an improvement. Observe that the value of σ changes in the transition from X_{kl-1} to X_{kl}, when a new leaf is reached. Hence not all pairs (σ, ν) are possible for X_n : when ν is a leaf node, then σ is necessarily the corresponding symbol. In other words, knowledge of ν is useless when a new symbol terminates. We denote such terminal states by (σ, ν_0), where ν_0 is the root node of the tree. As a result, the state space $\mathcal{X}$ for X is the product $\mathcal{S} \times \mathcal{T}$ where $\mathcal{T}$ is the set of inner vertices of the dyadic tree.

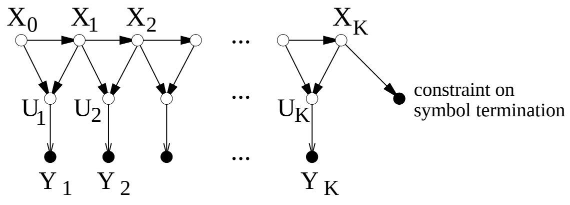


Figure 5: *Bayesian network of a bit clock model for the pair MS+SC, in the case of a constant length source code.*

The resulting Bayesian network of fig. 5 is again amenable to algorithms of section 3. The state space is larger : $|\mathcal{S} \times \mathcal{T}| \approx |\mathcal{S}|^2$, which requires more memory and means that a blackbox algorithm for this model will result in complexity $\mathbf{C'}_2 = O(N \cdot |\mathcal{S}|^2 \cdot |\mathcal{T}|^2) \approx O(N \cdot |\mathcal{S}|^4)$. But the complexity cannot be evaluated so loosely, because the transition matrix is very sparse. Let us consider l steps together ; with a rapid evaluation, from each (σ, ν_0) there are $|\mathcal{S}|$ decisions to take to reach the next symbol, which yields complexity $\mathbf{C}_2 = O(K \cdot |\mathcal{S}|^2)$. Hence a *careful* handling of this product model for MS+SC reaches the same performance as an estimation algorithm for MS alone.

4.1.3 Mixed clocks model

Following ideas that appeared in iterative decoding, one could imagine estimating the symbol stream by using the two models MS and SC alternatively, provided an interleaver is inserted between the source and the source coder. This idea is misleading here: we now prove that performing first a CLC decoding and injecting its soft output into an estimation algorithm for the MS model is an optimal strategy. This is due to the pointwise translation of symbols into codewords.

Let us reconsider the symbol clock model (fig. 3). Algorithm $\mathcal{A}$ applied to S corresponds to[7]

1. computing posterior marginals $\mathbb{P}(S_k|\bar{Y}_k)$ (or the ratio $\mathbb{P}(S_k|\bar{Y}_k)/\mathbb{P}(S_k)$), and

2. using these quantities as input of a two sweep procedure yielding $\mathbb{P}(S_k|Y)$.

As mentioned earlier, operation 1 amounts to estimating S, i.e. computing $\mathbb{P}_i(S_k|Y)$, assuming symbols are independent (whence notation $\mathbb{P}_i$), and operation 2 is in charge of incorporating the extra knowledge on inter-symbol correlation.

Although the symbol clock model looks natural for these two operations, it may be interesting to perform the first one with a bit clock model. This first operation corresponds to estimating $\bar{U}$ assuming $\mathbb{P}_i(\bar{U}) = \prod_k \mathbb{P}(\bar{U}_k)$, which can be done as in the previous section, on a bit clock basis. Symbol independence brings some simplifications into the picture: the state variable X_n doesn't need any more to keep track of the last symbol produced. So X_n reduces to a simple vertex ν of the dyadic tree, and the connection of local models is done with solution A. One point remains to be solved : how to get soft information on S from this model ? The solution is straightforward: there is no need to estimate bits U_n nor codewords $\bar{U}_k$, it suffices to estimate the state process X. Indeed, $\mathbb{P}_i(X_{kl}|Y)$ corresponds to the desired $\mathbb{P}_i(S_k|Y)$ since the possible values for X_{kl} are the leaves of the dyadic tree, i.e. possible symbols at time k. Hence the translation is immediate. By the way, notice that $\mathbb{P}_i(X_{kl}|Y)$ is readily obtained with one single sweep, since symbol independence induce $\mathbb{P}_i(X_{kl}|Y) = \mathbb{P}_i(X_{kl}|Y_1^{kl})$.

Operation 1 in algorithm $\mathcal{A}$ relies only on the inner codeword redundancy. If a (constant length) Huffman code is used, this represents at most 1 bit of redundancy per codeword, which is quite low. Anyway, the interest of algorithm $\mathcal{A}$ is to separate the use of the SC model and of the MS model. The SC model incorporates information on the *structure* of codewords (constraints in some sense), and is used to translate soft information from the bit clock to the symbol clock. The MS model incorporates the major part of source redundancy.

In terms of complexity, operation 1 amounts to estimating $\mathbb{P}(S_k|\bar{Y}_k)$, so its complexity $O(K \cdot |\mathcal{S}|)$ is negligible compared to the complexity of operation 2, $\mathbf{C}_3 = O(K \cdot |\mathcal{S}|^2)$.

[7]MPM is assumed, but the argument remains valid with $\bar{\mathbb{P}}$.

4.2 Joint decoding of the complete chain

While the SC removes some redundancy of the source (generally the intra-symbol redundancy), the CC reintroduces redundancy in the bit stream. Hence the joint use of the two sources of information MS and CC must be done with care. In this paper we consider a systematic convolutional channel code which, we believe, captures the structural difficulty of the joint decoding problem. Small bloc-codes are easy to handle as far as they cover one (or an integer number of) source symbol(s), since the Bayesian network incorporating redundant bits R remains (close to) a tree. In the general case, and in particular after VLC coding, the inter-symbol correlation due to bloc-codes brings the same difficulties as convolutional codes, since we come up with a very loopy BN, as shown below.

We rely on a trellis representation for the channel code (which could also capture the case of bloc-codes), so the CC has a state-space representation, with X' as state variable. Without loss of generality, we assume a bit clock recursion for the state equation : the CC takes information bits one at a time and yields a number of redundant bits, possibly none. The Bayesian network incorporating the complete chain MS+SC+CC is depicted on fig. 6 : the top part represents the bit clock product model for MS+SC, and the bottom part represents the serial concatenation of a convolutional encoder. Variables of R are depicted as functions of the coder state X', but could as well be functions of state transitions. Pointwise measurements Y and Z on U and R are not represented for clarity.

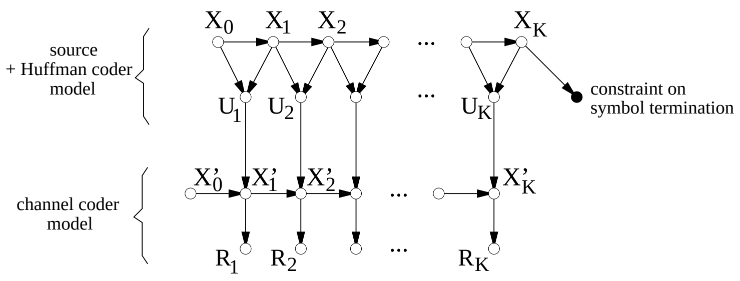


Figure 6: *Bayesian network of a bit clock model for the complete chain MS+SC+CC, in the case of a constant length source code. Measurements Y and Z are not represented for the sake of clarity.*

This BN is not a tree, hence algorithms of section 3, which are optimal for trees only, do not apply directly. A first solution to get back to a tree diagram is by means of "node aggregation." It consists in grouping nodes in order to remove all the cycles. Observe that the pair (X, X') forms a Markov process, and that (U_n, R_n) is a function of the transition from (X_{n-1}, X'_{n-1}) to (X_n, X'_n). So, by adopting this node aggregation, we are back to the standard framework of section 3. At the expense of a dramatic state augmentation however, since (X_n, X'_n) now gathers the last symbol of the source, the state of the source coder, and the state of the channel coder. This construction of a product model for the coding chain has been advocated in very simple cases by some authors [8], but is unaffordable in practical situations. A reasonable source alphabet satisfies $|\mathcal{S}| = 2^6$ or more, and usual convolutional coders need 5 bits memory, whence a state space dimension of 2^{17} or more...

An alternate solution to node aggregation is suggested by (serial) turbo codes. It was observed that the simple introduction of an interleaver between the pair MS+SC and the channel coder

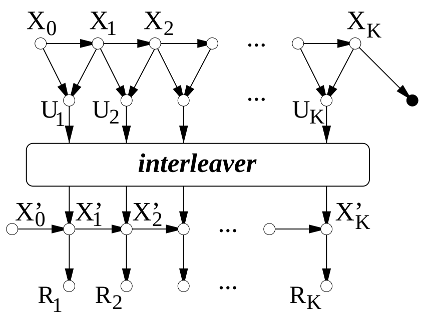


Figure 7: *Introduction of an interleaver to make short cycles long in the joint BN. Each R_n may contain 0,1 or several redundant bits.*

makes short cycles of the BN become long (fig. 7). A graphical model containing only long cycles can be locally approximated by a tree, around a given variable, taking into account that the correlation decays exponentially fast with the distance. Therefore, estimation algorithms designed for trees give good approximations on graphs with long cycles.

The easiest strategies to implement on a BN with loops belong to the "graph blind" family, as far as there is no obvious organization of message circulations. Updating all edge messages at a time amounts to collecting measurements lying at distance 1, then 2, and so on around each node, which provides a simple way to tune the tree approximation by defining a horizon to measurements involved in the estimates. In the case of concatenated Markov models, it is generally preferred to organize computations, for matters of simplicity in the decoding. The usual (turbo) strategy follows an iterative scheme alternating the use of the two models. It completes message circulations in one model (two sweeps) before updating messages towards the other model, which is processed in the same way. This iterative procedure offers the advantage of isolating two soft decoders, which minimizes cultural changes with respect to the separate decoding approach, and allows some interpretation of the approximations made, as shown below.

4.2.1 Iterative scheme with two models

Here, we stick to this traditional architecture of turbo algorithms and design an iterative scheme alternating uses of the CC model, and of the joint MS+SC model.

We consider the MPM criterion and rely on fig. 1 to give a macroscopic view of the procedure. Ideally, the first step computes $Ext_U(Z|Y)$ using the CC model, assuming some distribution $\mathbb{P}^0$ on the input U. Observe that $Ext_U(Z|Y) = \mathbb{P}^0(U|Y,Z)/\mathbb{P}^0(U|Y)$ is insensitive to the choice of distribution $\mathbb{P}^0$ on U since $Ext_U(Z|Y) \propto \mathbb{P}(Y,Z|U)/\mathbb{P}(Y|U)$; hence $\mathbb{P}_0$ can be chosen so as to make U a white noise. The second step uses $Ext_U(Z|Y)$ in conjunction with Y, as input to an estimation of U based on the true distribution $\mathbb{P}(U)$, as described by the MS+CC model.

This picture suffers from a severe difficulty : $Ext_U(Z|Y)$ is too complex to be handled globally, because bits of U are correlated given Z. In particular, $Ext_U(Z|Y)$ couldn't be used as input to the Markov model of the pair MS+SC since it is not homogeneous to a pointwise measurement process on U. Therefore an approximation is made after the first step : one introduces the white noise approximation $\mathbb{P}^0(U|Y,Z) \approx \prod_n \mathbb{P}^0(U_n|Y,Z)$ which yields

$$Ext_U^0(Z|Y) \quad \propto \quad \frac{\prod_n \mathbb{P}^0(U_n|Y,Z)}{\mathbb{P}^0(U|Y)} = \prod_n \frac{\mathbb{P}^0(U_n|Y,Z)}{\mathbb{P}^0(U_n|Y_n)} = \prod_n Ext_{U_n}(Y,Z|Y_n) \tag{20}$$

This has a double advantage.

- First, each of the local extrinsic information $Ext_{U_n}(Y,Z|Y_n)$ can now be used as a local measurement on U_n, thus allowing the use of the joint Markov model for MS+SC to get $\mathbb{P}^1(U|Y,Z)$, an approximation of the true $\mathbb{P}(U|Y,Z)$.
- Secondly, the assumption of a white noise initial distribution $\mathbb{P}^0(U) = \prod_n \mathbb{P}^0(U_n)$ for U allows the use of the CC Markov model to obtain $Ext_{U_n}(Y,Z|Y_n)$.

Unfortunately, unlike the global extrinsic information, this local extrinsic information depends on the choice of $\mathbb{P}^0$: the better the neighbours of U_n are estimated, the better this extrinsic information on U_n. Hence one should inject as much prior information as possible in the $\mathbb{P}^0$ distribution, still keeping the white noise assumption. This suggests the use of $\mathbb{P}^1(U_n|Y,Z)$ for U_n, from which the effect of local measurements Y_n and $Ext^0_{U_n}(Y,Z|Y_n)$ must be removed. Let us define[8]

$$Ext^1_{U_n}(Y|Y_n) \quad \propto \quad \frac{\mathbb{P}^1(U_n|Y,Z)}{\mathbb{P}(Un|Y_n)\cdot Ext^0_{U_n}(Y,Z|Y_n)} \tag{21}$$

We replace the prior $\mathbb{P}^0$ at the input of the CC decoder by the new prior $\mathbb{P}^2$

$$\mathbb{P}^2(U) \quad \propto \quad \prod_n \mathbb{P}(U_n)\cdot Ext^1_{U_n}(Y|Y_n) \tag{22}$$

which closes the first loop of the iterative procedure.

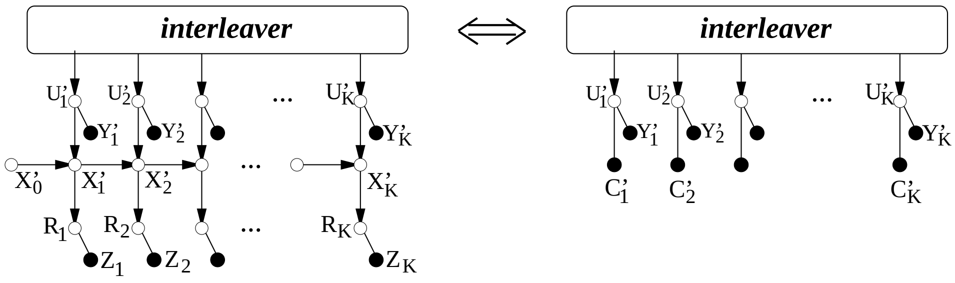


Figure 8: *Computing extrinsic information with the CC model. Primed letters U', Y' and $E^{0'}$ stand for interleaved versions of U, Y and E^0. E^0_n represents $Ext^0_{U_n}(Y,Z|Y_n)$. The equivalence sign means that the posterior distribution on U_n given Y_n and E^0_n is identical to the posterior distribution given Y and Z and based on the CC model.*

Once again, this architecture rephrases the turbo algorithm for serial turbo codes, and is nothing more than one possible organization of message circulations on the (loopy) BN of the joint model. Figures 8 and 9 illustrate the two steps of one iteration, incorporating also the interleaver. Local extrinsic informations are represented as grey patches, indicating they behave as pointwise measurements. Fig. 8 represents the result of soft decoding with the CC model, $Ext^0_{U_n}(Y,Z|Y_n)$ appears as a grey square close to each U_n. Fig. 9 depicts the second step, making use of the MS+SC Markov model, with Y_n and $Ext^0_{U_n}(Z|Y)$ as local measurements. The resulting extrinsic information $Ext^1_{U_n}(Y|Y_n)$ is represented as a grey triangle close to each U_n.

[8]Finding a good notation is difficult for this quantity, that depends both on Y and Z. The correct notation should be $Ext^1_{U_n}(Y, Ext^0|Y_n, Ext^0_{U_n})$. We choose to focus on the dependence on Y.

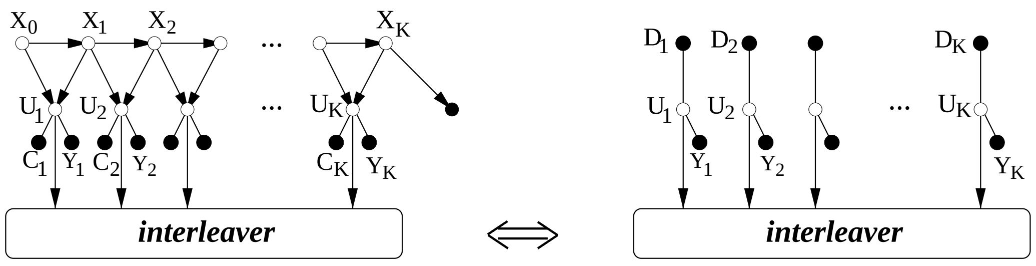


Figure 9: *Computing extrinsic information with the MS+SC model.* E_n^1 *represents* $Ext^1_{U_n}(Y|Y_n)$*. The equivalence sign means that the posterior distribution on* U_n *given* Y_n, E_n^0 *and* E_n^1 *is identical to the posterior distribution given* Y *and* E^0 *and based on the MS+SC model.*

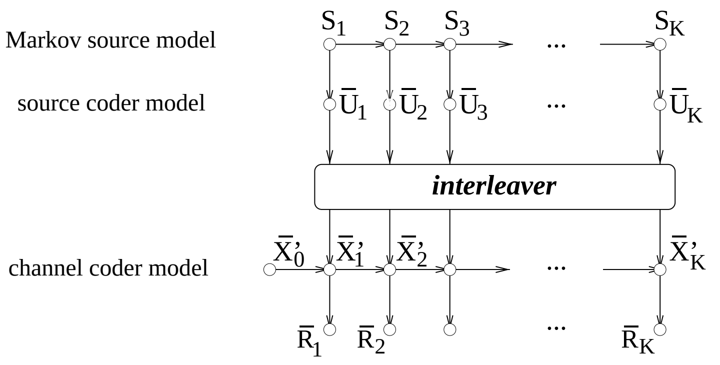
Remarks.

- At the second step of each iteration (fig. 9), one needs the posterior distribution on U_n, which follows from the posterior distribution on pairs (X_{n-1}, X_n). This increases the computational complexity and is in favor of a small state space $\mathcal{X}$.
- Stopping the algorithm at the first iteration amounts to performing a soft channel decoding followed by source decoding.
- At the last iteration, one should not keep an MPM estimate of the bit stream U, since a bit by bit MPM estimation may very well yield non valid codewords. Instead, the MPM estimates of symbols S_k must be read out of the MPM estimate of the state process X : $\hat{X}_{kl}$ gives $\hat{S}_k$.
- Finally, notice that the final MPM estimation of the useful bitstream U with the MS+SC model can be replaced by a MAP estimation, considering extrinsic information as extra measurements. The MAP estimate of the bitstream U necessarily gives valid codewords, since it corresponds to the MAP estimate of the symbol stream S.

4.2.2 Iterative scheme with three models

With no additional complexity, the decoding of the MS+SC pair (second step of each iteration) can be performed with the mixed clocks model. This amounts to estimating the bit stream U using first the intra-codeword redundancy (i.e. the SC model alone), exactly as above (section 4.2.1). More precisely, the state process X of the SC model is estimated. Then the symbol stream can in turn be estimated using the inter symbol correlation (i.e. the MS model), as was shown in section 4.1.3. To prepare for the next iteration, the resulting posterior distributions $\mathbb{P}^1(S_k|Y,Z)$ must be transformed back into $\mathbb{P}(U_n|Y,Z)$, which is the only novelty. This "clock conversion" is straightforward, as shown in section 4.1.2, and is much simpler than in section 4.2.1 since no posterior distribution on pairs (X_{n-1}, X_n) is necessary (see the first remark there).

This last approach results in a completely separated use of the three models in the chain, provided one interleaver is introduced. We have chosen an architecture where the interleaver is placed between the SC and the CC. Notice that the same approach remains valid with the interleaver placed between the MS and the SC. This requires to design a symbol clock model of the CC, which can be done by aggregating l consecutive states of the CC (fig. 10). The advantage lies in a better white noise approximation $\mathbb{P}^0(U|Y,Z) \approx \prod_k \mathbb{P}^0(\bar{U}_k|Y,Z)$, whence a better treatment of the extrinsic information of the CC than in (20). This is done however at the expense of a more complex soft channel decoder.



Markov source model
source coder model
channel coder model
interleaver
S_1 S_2 S_3 ... S_K
$\bar{U}_1$ $\bar{U}_2$ $\bar{U}_3$ ... $\bar{U}_K$
$\bar{X}'_0$ $\bar{X}'_1$ $\bar{X}'_2$ ... $\bar{X}'_K$
$\bar{R}_1$ $\bar{R}_2$ ... $\bar{R}_K$

Figure 10: *With constant length source codes, the interleaver can be defined on symbols instead of bits, and placed either before or after the source coder.*

5 Joint source-channel decoding for variable length source codes

This section addresses the central purpose of the paper : the general case of VLC encoded sources. Lengths K and N, of the symbol stream and of the bit stream, are supposed to be known ; we will indicate how algorithms simplify when one of these information is missing. We focus on the joint decoding of the pair MS + SC, since the introduction of the CC follows the same lines as before : either an unrealistic product model is constructed, or an iterative approach is chosen. The difficulty of VLCs comes from the lack of synchronization between the symbol clock and the bit clock. In other words, the estimation of the transmitted bit stream must be performed jointly with its *segmentation.* This makes VLCs less robust to transmission noise, since more information must be recovered for their decoding. To estimate the segmentation of the received bit stream into codewords, one must determine the value K_n of the symbol clock at each bit instant n, when a bit clock model is used for estimation. Conversely, when a symbol clock model is used, one must determine the value N_k of the bit clock at each symbol instant k. We therefore review the models developed in the previous section in order to introduce these clock variables, and show how algorithms adapt.

5.1 Symbol clock model

Let us define N_k as the number of bits in the VLC coding of symbols $S_1 \dots S_k$. Starting with $N_0 = 0$, one has the recursion

$$N_k \quad = \quad N_{k-1} + \mathcal{L}(S_k) \tag{23}$$

where $\mathcal{L}(S_k)$ is the length of the codeword $\bar{U}_k$ associated to S_k. We still assume S is a Markov chain, hence the extended process (S,N) composed of pairs (S_k, N_k) remains a Markov chain[9].

[9]With our conventions, notation N stands either for process (N_k) or for the length of the bitstream. But the context generally solves any ambiguity on the meaning of N.

The Bayesian network of fig. 3 transforms into the one of fig. 11, where the codeword $\bar{U}_k$ has been further expanded to display its internal bits $U_{N_{k-1}+1} \dots U_{N_k}$. The similarity is both inspiring and misleading. Apparently one still has a tree shaped graphical representation, which is favorable to estimation algorithms. But a closer look reveals that the tree structure is *random*! Indeed, the connection from (S_k, N_k) to $\bar{U}_k$ is actually a connection to a variable number of bits, at a variable position in the bitstream... Therefore the tree structure varies with the values of process (S, N).

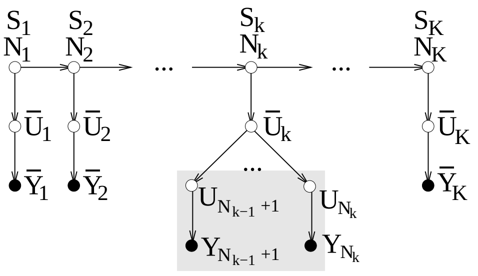


Figure 11: *Bayesian network of a symbol clock model for the pair MS+SC, in the case of a variable length source code. The tree-shaped BN is random: connections depend on values of S.*

Nevertheless, we now demonstrate a striking property: *the estimation of (S, N) can be performed exactly as before, just ignoring this random BN phenomenon.* More precisely, the organized estimation strategies of section 3.1.1 remain valid. For simplicity, we do not develop a full theoretical argument, which would go beyond the scope of this paper. Instead, we show that factorization (2) still holds under the form

$$\mathbb{P}(S_k, N_k | Y) \quad \propto \quad \dot{\mathbb{P}}(S_k, N_k | \bar{Y}_1^k) \cdot \mathbb{P}(\bar{Y}_{k+1}^K | S_k, N_k) \tag{24}$$

where $\dot{\mathbb{P}}$ is defined below, from which one can easily check that all other formulae of section 3.1.1 hold also. Quotation marks should appear around the right-hand side term, to indicate that a quantity like $\dot{\mathbb{P}}(S_k, N_k | \bar{Y}_1^k)$ is not a correct conditional distribution on (S_k, N_k): the conditioning variable $\bar{Y}_1^k = Y_1 \dots Y_{N_k}$ varies with values of (S_k, N_k)... However, handling this object as a regular conditional distribution allows a correct computation of $\mathbb{P}(S_k, N_k | Y)$, which is properly defined.

Let us start with the definition of $\mathbb{P}(\bar{Y}_{k+1}^K | S_k, N_k)$. The randomness of the tree structure is due to S. Given process S, and thus process N, the tree structure is fixed, so we have

$$\begin{aligned}
\mathbb{P}(Y, S_k, N_k) &= \mathbb{P}(Y | S_k, N_k) \cdot \mathbb{P}(S_k, N_k) && (25)\\
&= \mathbb{P}(Y_1 \dots Y_{N_k} | S_k, N_k) \cdot \mathbb{P}(Y_{N_k+1} \dots Y_N | S_k, N_k) \cdot \mathbb{P}(S_k, N_k) && (26)\\
&= \mathbb{P}(\bar{Y}_1^k | S_k, N_k) \cdot \mathbb{P}(\bar{Y}_{k+1}^K | S_k, N_k) \cdot \mathbb{P}(S_k, N_k) && (27)
\end{aligned}$$

Equation (27) holds because for every value (s_k, n_k) of the pair (S_k, N_k), vectors $\bar{Y}_1^k = Y_1 \dots Y_{N_k}$ and $\bar{Y}_{k+1}^K = Y_{N_k+1} \dots Y_N$ are perfectly defined, and are conditionally independent. Hence the required conditional distributions on $\bar{Y}$ given (S_k, N_k) are properly defined. We can further define

$$\mathbb{P}(\bar{Y}_1^k, S_k, N_k) \quad = \quad \mathbb{P}(Y_1 \dots Y_{N_k} | S_k, N_k) \cdot \mathbb{P}(S_k, N_k) \tag{28}$$

which sums to 1 as an ordinary distribution, as far as we start summations by the Y part.

Pushing further the procedure to define $\mathbb{P}(\bar{Y}_1^k)$ and then $\mathbb{P}(S_k, N_k|\bar{Y}_1^k)$ fails. There is no simple way to give a meaning to $\mathbb{P}(\bar{Y}_1^k)$ by summing over (S_k, N_k), because the very definition of $\bar{Y}_1^k = Y_1 \dots Y_{N_k}$ requires the knowledge of N_k. We proceed in another way. Equations (27) and (28) yield

$$\mathbb{P}(S_k, N_k, Y) \quad = \quad \mathbb{P}(S_k, N_k, \bar{Y}_1^k) \cdot \mathbb{P}(\bar{Y}_{k+1}^K | S_k, N_k) \tag{29}$$

In practice, one sample y of the measurement process Y is available, as input to the estimation algorithm, and $\mathbb{P}(S_k, N_k|Y = y)$ is obtained by renormalizing $\mathbb{P}(S_k, N_k, Y = y)$. Let us define $\dot{\mathbb{P}}(S_k, N_k|\bar{Y}_1^k)$ *for that particular value* y as

$$\dot{\mathbb{P}}(S_k, N_k|\bar{Y}_1^k) \quad \propto \quad \mathbb{P}(\bar{Y}_1^k = \bar{y}_1^k, S_k, N_k) \tag{30}$$

where $\propto$ stands for a renormalization over (S_k, N_k). Then (24) holds for that particular value y, which is all we need in practice for estimating (S, N). One easily checks that recursions of section 3.1.1 also hold[10] with this definition of $\dot{\mathbb{P}}$.

As a consequence, symbol clock based decoding algorithms developed for CLC encoded sources remain valid for VLCs. The only (light) difference lies in the computation of $\mathbb{P}(\bar{Y}_k|S_k, N_k)$, which requires to pick measurements at the right place in the received bitstream. A direct computation of $\mathbb{P}(\bar{Y}_k|S_k, N_k)$ is possible but seems inappropriate because it doesn't follow the natural time index of measurements, which is the bit clock. An alternate solution is proposed in section 5.4, with similar complexity.

Soft information on bits. The use of a symbol clock model for iterative decoding of the chain MS+SC+CC requires the translation of posterior marginals $\mathbb{P}(S_k, N_k|Y)$ into posterior marginals $\mathbb{P}(U_n|Y)$. This point is developed in subsection 5.4.

Constraints on the number of bits/symbols. An estimation algorithm based on the symbol clock model defined so far yields an optimal sequence of pairs (S_k, N_k). In other words, the best sequence of K symbols is chosen regardless of its length in number of bits. The easiest way to incorporate knowledge on the number of bits is to add one extra measurement node on the last pair (S_K, N_K) stating that N_K equals the required number of bits. This measurement node is particular since there is no observation noise in the measurement ; it actually encodes a constraint. Another strategy consists in computing a model for process (S, N) conditionally to the fact that the last value N_K is the required number of bits. It can be shown that this model is still a Markov chain, but homogeneity is lost.

When the number of bits is known, and the number of symbols is left free, the Markov model on process $(S_k, N_k)_{k=1\dots K}$ must be modified. First, K must be large enough to allow all symbol sequences of N bits. Then, once N_k reaches the required length, the model must enter and remain in a special state for which all future measurements are non-informative.

[10]No big mystery in the above developments : estimation algorithms rely on the factorization properties of the distribution $\mathbb{P}(S, N, Y)$; renormalizations appearing in computations are generally harmless and mainly meant to favor the stability of algorithms. Hence a proper definition of conditional probabilities is useless.

5.2 Bit clock model

The procedure to build a bit clock model for the pair MS+SC follows the lines of section 4.1.2. We have factorization $\mathbb{P}(U) = \prod_k \mathbb{P}(\bar{U}_k|\bar{U}_{k-1})$, the terms of which must be decomposed further to display a bit clock recursion.

The decomposition of each term $\mathbb{P}(\bar{U}_k|\bar{U}_{k-1})$ can be done as before, by mapping this conditional distribution on the tree representation of codewords. In usual entropic coding schemes, it corresponds to a Huffman tree for the stationary distribution of the symbol source. Again, this yields a stochastic automaton construction of the kth codeword, which can be put in state space form. The state variable X_n is a pair (σ, ν), where σ is the last symbol produced, and ν a vertex of the codeword tree. By contrast with the fixed length case, knowing the bit index n is not sufficient to determine the rank k of the symbol being constructed, i.e. to determine what probability $\mathbb{P}(\bar{U}_k|\bar{U}_{k-1})$ governs the next transition. Therefore this information must be available jointly with the state variable X_n. Let us denote by K_n the number of achieved symbols at time n. The connection of local models now amounts to defining a Markov chain distribution on pairs (X_n, K_n). For what concerns the X_n part, trees are connected one to the other like in the CLC case (section 4.1.2), with either solution A or solution B. The transition probability from X_n to X_{n+1} is thus determined by $\mathbb{P}(\bar{U}_{K_n+1}|\bar{U}_{K_n})$. For the K_n part, $K_{n+1} = K_n + 1$ each time a new symbol is achieved by X_n, i.e. each time X_{n+1} reaches a new leafnode, otherwise $K_{n+1} = K_n$.

To ensure that the last bit of the chain terminates a symbol, an extra measurement (or constraint) node can be added on the last state (X_N, K_N). This measurement μ takes the form $\mathbb{P}(\mu|\sigma, \nu, k) = 1$ if σ is a leafnode, and 0 otherwise (normalization is useless).

In terms of complexity, the symbol clock model and the bit clock model are equivalent, as in the CLC case, provided the sparse transition matrix of the latter is handled properly. This will become clear in the next section. Again, if black box algorithms are used, the state space size favors the symbol clock model. Notice also that extracting posterior marginals $\mathbb{P}(U_n|Y)$ requires the transition posterior marginals $\mathbb{P}(X_{n-1}, K_{n-1}, X_n, K_n|Y)$, which is a large object and penalizes this model.

Constraints on the number of bits/symbols. The bit clock model as defined above imposes no constraint on the number of symbols. The symbol counter K_n only helps selecting the right transition probability on symbols. So when S is a stationary Markov chain, K_n becomes useless and can be removed. An estimation algorithm based on this model will yield the best decoding of the sequence of N bits, regardless of the number of symbols.

If the number of symbols is known, this information can be incorporated as before, by adding an extra measurement on the last state (X_N, K_N) of the bit clock model, constraining the value of K_N.

Symmetrically to the symbol clock model, for the case of a fixed number of symbols and a free number of bits, the model must be modified. N must be set large enough to capture the longest bit stream. And when the right number of symbols is reached in some (X_n, K_n), the model must enter and remain in a special state for which all future measurements are non-informative.

5.3 Trellis for joint decoding

The relationship between the symbol clock model and the bit clock model is best evidenced using a global trellis representation, as suggested in [12, 13]. Actually, the trellises of the two models are almost identical ; models essentially differ by the definition of state variables, that involve different cuts of the trellis.

Let us define a *state* of the global trellis as a 4-tuple (σ, ν, k, n) where

- $\sigma \in \mathcal{S}$ is the last completed symbol,
- $\nu \in \mathcal{T}$ is a vertex of the Huffman tree,
- k is the number of completed symbols, and
- n is the length of the bit stream up to that state.

Hence the two clock indexes appear in this definition of a state. Transitions are defined in the following way (we adopt solution A for connecting successive codeword trees). Let $s = (\sigma, \nu, k, n)$ and $s' = (\sigma', \nu', k', n')$ be two states, a transition from s to s' is possible iff

1. $n' = n + 1$,
2. ν' is a leafnode $\Rightarrow$ $k' = k + 1$, otherwise $k' = k$,
3. ν' is a leafnode $\Rightarrow$ σ' is the corresponding symbol, otherwise $\sigma' = \sigma$,
4. ν' is a successor of ν on the codeword tree, or ν is a leafnode and ν' an immediate successor of the rootnode.

Rule 4 makes each terminal leafnode of a codeword tree the rootnode of the next tree (solution A). Transition $s \to s'$ obviously produces bit U_{n+1} which belongs to symbol S_{k+1}, given that the k-th symbol was σ. Hence the transition likelihood is determined by the $\nu \to \nu'$ transition on the codeword tree, equipped with $\mathbb{P}(S_{k+1} | S_k = \sigma)$ (again, if ν is a leafnode, it must be read as the rootnode).

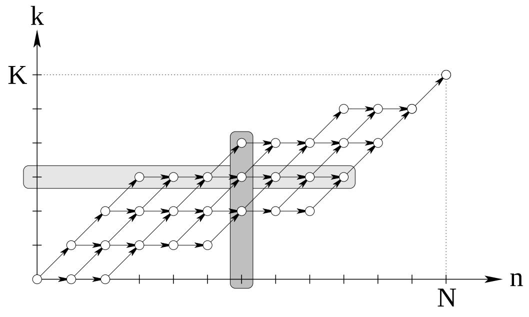


Figure 12: *Global trellis of the pair MS+SC, constrained to produce K symbols on N bits. Only the (k, n) part of states is represented. Vertical cuts define the bit clock model, and horizontal cuts (after some state elimination) define the symbol clock model.*

Figure 12 gives an example of a global trellis for the pair MS+SC constrained to produce K symbols in a length N bit stream. Only the (k, n) part of states is represented for clarity, for codeword lengths varying between 1 and 3 bits. Let us recall some obvious properties of trellises. Each path from the initial state (bottom-left) to a final state (top-right) corresponds to one possible sequence of symbols and bits. The probability of a path is naturally the product of all transition probabilities, and over all possible paths that probability sums to 1. The probability of a state is the sum of path probabilities over all paths going through that state. States can be removed from the trellis, for a model reduction purpose, for example. When s is removed, each predecessor s^- must be connected to all successors s^+, and path probabilities follow accordingly. A *cut* of the trellis is a set of states such that none of them is a successor of another, and such that every path of the trellis goes through one (unique) state of that set. As a consequence, the sum of state probabilities over a cut is 1. Successive cuts allow to define state variables, and consequently a state space representation of the trellis. For example, states $s = (\sigma, \nu, k, n)$ with the same value n form a cut at bit time n. Let us define ξ_n as the random variable taking values in that cut, then one recovers the bit clock model in the Markov chain $(\xi_n)_{n=0\ldots N}$. The symbol clock model can be recovered in a similar manner. Let us first remove all states $s = (\sigma, \nu, k, n)$ for which ν is an internal node of the codeword tree. Then only states corresponding to leaf nodes, i.e. symbols, remain in the trellis. Notice that σ and ν represent the same symbol in the remaining states, hence the state space dimension can be reduced. On that transformed trellis, states with identical symbol clock value define cuts corresponding to the symbol clock model.

Each transition $s \to s'$ of the trellis produces one bit, say $U_n = u_n$, and thus is associated to one measurement, $Y_n = y_n$. Let us multiply the transition probability by the conditional likelihood $\mathbb{P}(Y_n = y_n | U_n = u_n)$. Then a MAP estimation amounts to computing the best path of the trellis for the new transition costs. By contrast, MPM estimation computes the probability of each state, for this modified transition probability, performing the adequate renormalizations in successive cuts. It becomes quite obvious on this representation that the bit clock model and the symbol clock model require the same amount of computations. However, states are smaller for the latter, at the expense of a dense transition matrix, and states are larger for the former, with a sparse transition matrix. If this sparsity is ignored, complexity augments dramatically.

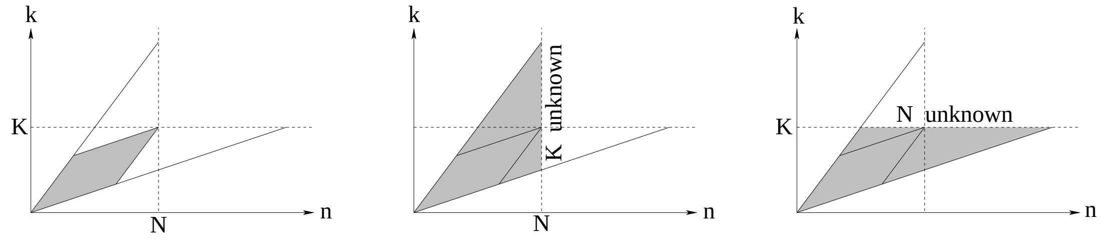


Figure 13: *Range of possible states of the trellis when K or N is left free. The slopes of boundaries are determined by the shortest and longest codeword.*

As mentioned in the previous sections, the constraint on the number of symbols/bits can be placed on the last state of the model. However, this requires visiting much more states than necessary (see fig. 13) : for example, with the bit clock model, all states with a wrong number of symbols will be discarded at time N. An alternate solution consists in taking as prior a Markov model *including* the constraint. Here, the drawback is the inhomogeneity of the model. The easiest solution is a mixture : it keeps the original unconstrained model, but doesn't visit states that will not satisfy the final constraint. This is harmless since such states have a null contribution to the result.

5.4 Mixed clocks model

Following ideas developed for the CLC case, we now consider an estimation strategy using separately the MS model and the SC model. It offers the advantage of requiring smaller state vectors, in particular for the bit clock model of the source coder. Surprisingly, algorithm $\mathcal{A}$ remains valid with a VLC, up to some technical modifications.

Let us recall the principle for the CLC case. The point is to get $\mathbb{P}(S_k|\bar{Y}_k)$ or more precisely $\mathbb{P}(\bar{Y}_k|S_k)$, which serves as input to an estimation algorithm for the MS alone (symbol clock). This quantity comes as a byproduct of the forward sweep of an MPM algorithm for the SC model (bit clock), assuming symbols are independent: one has $\mathbb{P}_i(S_k|\bar{Y}_1^k) = \mathbb{P}(S_k|\bar{Y}_k)$, whence $\mathbb{P}(\bar{Y}_k|S_k) \propto \mathbb{P}_i(S_k|\bar{Y}_1^k)/\mathbb{P}(S_k)$.

Symmetrically, in the VLC case the point is to get $\mathbb{P}(\bar{Y}_k|S_k, N_k)$ and use it as input to a two-sweep estimation algorithm on the MS model (section 5.1). We now show how $\mathbb{P}(\bar{Y}_k|S_k, N_k)$ derives from the forward sweep of an MPM algorithm on the SC model fed with independent symbols.

How to obtain $\mathbb{P}(\bar{Y}_k|S_k, N_k)$. Let us assume a source of independent symbols $\mathbb{P}_i(S) = \prod_k \mathbb{P}(S_k)$. As shown in section 5.1, the model must be augmented with a bit counter $N_k = \mathcal{L}(S_1^k)$ satisfying (23). This counter is necessary to recover symbol boundaries in the bit stream, and to identify measurements associated to a given symbol S_k. Notice that the augmented process $(S_k, N_k)_{k=1\ldots K}$ is *not* a white noise anymore, but becomes a Markov chain, precisely because of recursion (23). Its transition probability is given by

$$\mathbb{P}_i(S_k, N_k|S_{k-1}, N_{k-1}) \;=\; \mathbb{P}(S_k) \cdot \mathbb{I}_{N_k=N_{k-1}+\mathcal{L}(S_k)} \tag{31}$$

A forward sweep of an MPM algorithm on this symbol clock model produces $\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k)$ for all values of k. A "backward reading" of recursion equations provide a way of extracting $\mathbb{P}(\bar{Y}_k|S_k, N_k)$. The update equation (3) becomes

$$\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k) \;\propto\; \dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^{k-1}) \cdot \mathbb{P}(\bar{Y}_k|S_k, N_k) \tag{32}$$

whence the desired $\mathbb{P}(\bar{Y}_k|S_k, N_k)$ by

$$\mathbb{P}(\bar{Y}_k|S_k, N_k) \;\propto\; \frac{\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k)}{\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^{k-1})} \tag{33}$$

The denumerator derives from the modified propagation (or prediction) equation (4)

$$\begin{aligned}
\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^{k-1}) &= \sum_{s_{k-1},n_{k-1}} \dot{\mathbb{P}}_i(s_{k-1}, n_{k-1}|\bar{Y}_1^{k-1}) \cdot \mathbb{P}_i(S_k, N_k|s_{k-1}, n_{k-1}) && (34)\\
&= \sum_{s_{k-1},n_{k-1}} \dot{\mathbb{P}}_i(s_{k-1}, n_{k-1}|\bar{Y}_1^{k-1}) \cdot \mathbb{P}(S_k) \cdot \mathbb{I}_{N_k=n_{k-1}+\mathcal{L}(S_k)} && (35)\\
&= \mathbb{P}(S_k) \cdot \dot{\mathbb{P}}_i(N_{k-1}|\bar{Y}_1^{k-1})\Big|_{N_{k-1}=N_k-\mathcal{L}(S_k)} && (36)
\end{aligned}$$

$\mathbb{P}(S_k)$ is given and $\dot{\mathbb{P}}_i(N_{k-1}|\bar{Y}_1^{k-1})$ comes from $\dot{\mathbb{P}}_i(S_{k-1}, N_{k-1}|\bar{Y}_1^{k-1})$, which concludes the first point.

How to obtain $\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k)$. Of course, the interest of the method is to determine $\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k)$ with the bit clock model of the SC alone: this is the most natural manner of recursively introducing measurements Y_n.

As in the CLC case, when symbols are independent, the SC state variable is still a pair (X_n, K_n), but X_n doesn't need any more to keep track of the last symbol produced, hence it reduces to the ν part. The global trellis associated to this model reproduces the one of section 5.3, except for the σ component of states, which disappears.

A complete MPM algorithm on this trellis computes the posterior probability of each state given all measurements: $\mathbb{P}_i(\nu, k, n|Y)$. This posterior probability sums to 1 over *all* cuts of the trellis. However, we actually need the result of the *forward* sweep only, for which this nice normalization result doesn't hold. Let us assume that an MPM algorithm is run over the trellis without performing any renormalization. At the end of the forward sweep one gets $\mathbb{P}_i(\nu, k, n, Y_1^n)$. The distribution $\dot{\mathbb{P}}_i(S_k, N_k|\bar{Y}_1^k)$ we are looking for corresponds to $\mathbb{P}_i(\nu, k, n, Y_1^n)$ normalized over the *horizontal* cut $\mathcal{C}_k^H$ defined by k fixed, n free and ν a leafnode of the codeword tree (see fig. 12). But a true MPM algorithm, assuming a recursion on n, recursively normalizes $\mathbb{P}_i(\nu, k, n, Y_1^n)$ over the *vertical* cut $\mathcal{C}_n^V$ at bit time n, defined by n fixed, k free and ν free. Therefore either vertical normalizations are removed (which does not favor the stability of the MPM procedure) or the successive vertical renormalization factors must be stored, in order to renormalize correctly the horizontal cuts $\mathcal{C}_k^H$.

Soft information on U_n : $\mathbb{P}(U_n|Y)$. The use of a symbol clock model for iterative decoding of the chain MS+SC+CC requires the translation of posterior marginals $\mathbb{P}(S_k, N_k|Y)$ into posterior marginals $\mathbb{P}(U_n|Y)$. To explain this procedure, we rely on the trellis representation of section 5.3. Conditionally to measurements Y and to the termination constraints, the process (S, N) still has a Markov chain structure. In other words, the conditional law $\mathbb{P}(S, N|Y)$ can be expanded on the global trellis with states (σ, ν, k, n). Let us denote by $\mathbb{P}_{|Y}$ the posterior distribution on (S, N). Then $\mathbb{P}_{|Y}(U_n = u, K_n = k)$ is determined by the probability of transitions $(\sigma, \nu, k, n-1) \rightarrow (\sigma', \nu', k', n)$ producing bit value u at bit time n and inside the k-th symbol. Since the bit value produced by transition $s \rightarrow s'$ on the trellis depends only on the corresponding transition $\nu \rightarrow \nu'$ on the codeword tree, one has

$$\begin{aligned} & \mathbb{P}_{|Y}(U_n = u, K_n = k) \\ = & \sum_{\sigma,\sigma',\nu,\nu' \,:\, (\nu \rightarrow \nu') \Rightarrow u} \mathbb{P}_{|Y}(\sigma, \nu, k, n-1) \cdot \mathbb{P}_{|Y}(\sigma', \nu', k', n|\sigma, \nu, k, n-1) \quad (37) \\ = & \sum_{\nu,\nu' \,:\, (\nu \rightarrow \nu') \Rightarrow u} \mathbb{P}_{|Y}(\nu, k, n-1) \cdot \mathbb{P}_{|Y}(\nu', k', n|\nu, k, n-1) \quad (38) \end{aligned}$$

Equation (38) expresses that $\mathbb{P}_{|Y}(U_n = u, K_n = k)$ requires only a compressed version of the global trellis, where memory of the last symbol produced σ has been removed. This compressed trellis corresponds to replacing the true distribution $\mathbb{P}_{|Y}(S, N) = \prod_k \mathbb{P}_{|Y}(S_k, N_k|S_{k-1}, N_{k-1})$ by $\prod_k \mathbb{P}_{|Y}(S_k, N_k|N_{k-1}) = \prod_k \mathbb{P}_{|Y}(S_k|N_{k-1}) \cdot \mathbb{I}_{N_k = N_{k-1} + \mathcal{L}(S_k)}$. In other words, only the posterior marginals $\mathbb{P}_{|Y}(S_k, N_k)$ are necessary to determine $\mathbb{P}_{|Y}(U_n, K_n)$, and consequently $\mathbb{P}_{|Y}(U_n)$. To summarize, the translation of soft information on pairs (S_k, N_k) into soft information on U_n can be done by 1/ decomposing $\mathbb{P}_{|Y}(S_k, N_k)$ into $\nu \rightarrow \nu'$ transition probabilities, 2/ placing these probabilities on edges of the reduced trellis, composed of states (ν, k, n), and 3/ collecting transition probabilities corresponding to the production of bit U_n ("vertical" summations on fig. 12).

Complexity. In terms of complexity, results are similar to the case of CLCs. The total amount of computations are identical in the three strategies, up to some multiplicative constant. Basically, all vertices of the global trellis must be visited. The symbol clock model works on a reduced state space, but deals with a dense transition matrix, whereas the bit clock model uses a larger state space and involves a sparse transition matrix. Complexities remain similar as far as this sparsity is properly handled (which is not easy to implement). But the bit clock model is penalized for computing $\mathbb{P}(U_n|Y)$, since it requires posterior transition probabilities between large states. The mixed clocks model case reduces the overhead of soft information conversion, and preserves the advantage of a bit clock recursion for the introduction of measurements.

6 Experiments

To evaluate the performance of the joint decoding procedure, experiments have been performed on a first-order Gauss-Markov source, with zero-mean, unit-variance and correlation factor $\rho = 0.9$. The source is quantized with a 16 levels uniform quantizer (4 bits) on the interval $[-3, 3]$, and we consider sequences of $K = 200$ symbols. The VLC source coder is based on a Huffman code, designed for the stationary distribution of the source. The channel code is a recursive systematic convolutional code of rate 1/2, defined by the polynomials $F(z) = 1 + z + z^2 + z^4$ and $G(z) = 1 + z^3 + z^4$. Since very few errors have been observed with rate 1/2, we have augmented it to 3/4 by puncturing the redundant bit stream. A variable size interleaver is introduced between the source coder and the channel coder. All the simulations reported here have been performed assuming an additive white Gaussian channel with a BPSK modulation. The results are averaged over 500 channel realizations.

Figure 14 provides the residual bit error rates (BER) and symbol error rates (SER) for different channel E_b/N_0. On each plot, the top curve corresponds to an ML estimation of the bitstream assuming independent bits (and no channel coding), followed by a hard Huffman decoding. On the BER plot, the second curve corresponds to a MAP channel decoding, assuming an input of independent bits. The third one is the result of the first iteration, where knowledge on symbol correlation and codeword structure has been introduced. Successive curves show the extra gain of iterations in the procedure, which depends on the degree of redundancy present on both sides of the source coder (see the next experiment, assuming independent symbols). For a BER of 10^{-4}, the joint source-channel turbo decoding system based on the three models brings at the first iteration a gain of 1 dB over the classical MAP channel decoding (with rate 3/4). An additional gain of around 2.5 dB has been obtained between the first and the fourth iterations.

The same experiments have been performed assuming the symbol source is white (fig. 15, in order to evidence the gain introduced by the intersymbol correlation. On the BER plot, the top curve still represents the error rate without channel coding. The second one is obtained using the CC model only (1st step of the 1st iteration). Then comes the BER after the first iteration for a white noise model, which can be viewed as the BER at the output of the SC model for the Gauss-Markov source. And the lowest curve is the BER at the end of the first iteration for the Gauss-Markov source. Hence these four curves help understanding the effect of each component in the model. As expected, the SC model has little influence since it uses little bit correlation and mainly relies on constraints on the number of bits and on codeword structure. Nevertheless, this effect is sufficient enough to evidence some gain in the successive iterations, when symbols are assumed to be independent. A comparison with the Markov source case shows that taking the inter-symbol correlation into account brings a gain of more than 2 dB for the SER.

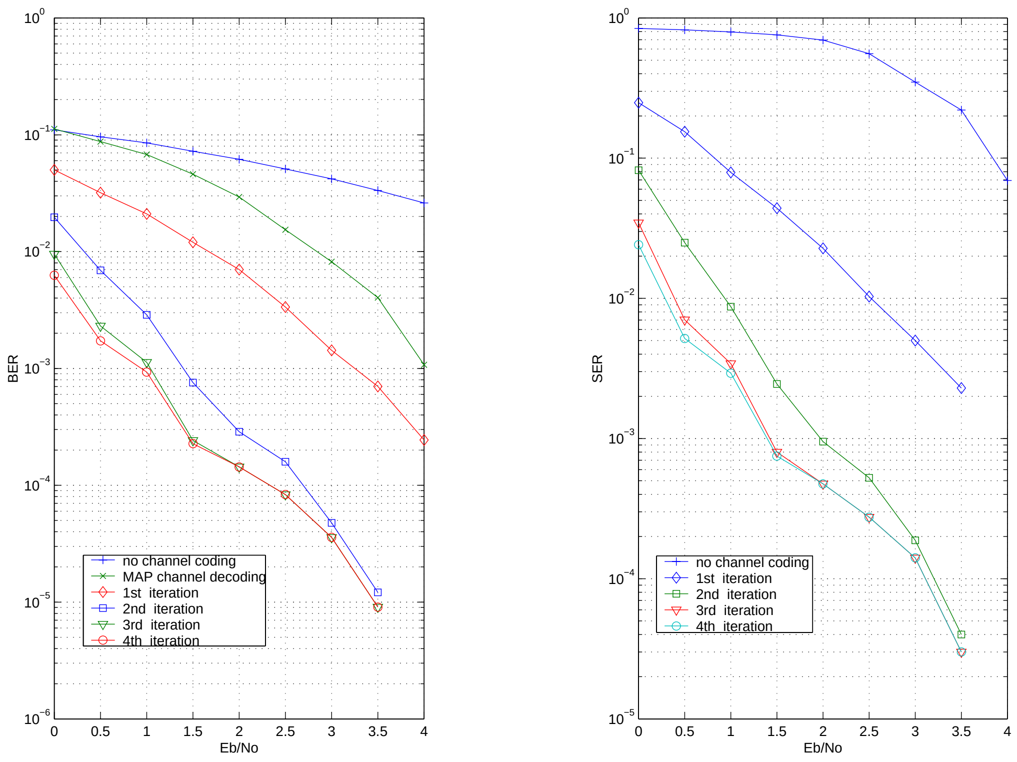
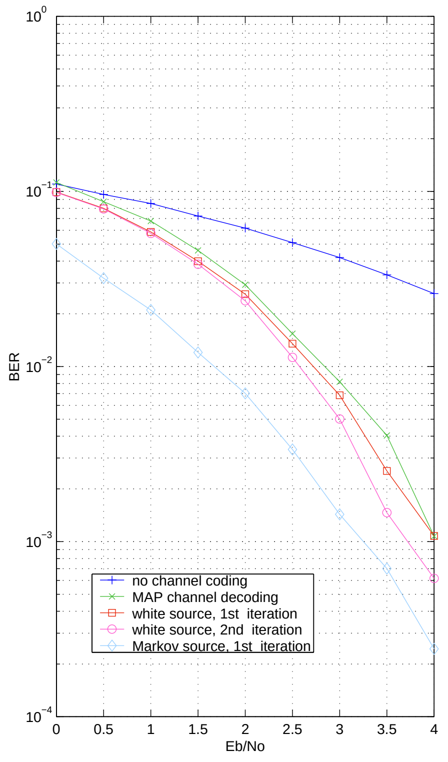


Figure 14: *Residual BER (left) and SER (right) for different* E_b/N_0*, for successive iterations (with a maximum of four iterations), for a Gauss-Markov source of 200 symbols quantized on 4 bits. The results are averaged over 500 AWGN channel realizations.*



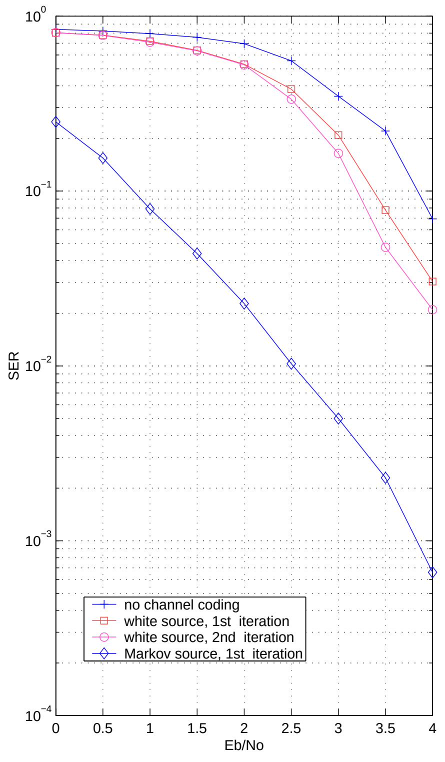


Figure 15: *Same conditions as the previous figure, except that inter-symbol correlation is not taken into account (a white source is assumed).*

The synchronization losses have also been estimated. This phenomenon is illustrated on fig. 16, for a sequence of 60 symbols. The top curve displays symbol lengths, alternating white and black patches. The upper sequence represents the estimated symbols, and the lower sequence the actual values. A desynchronization occurs at symbols 11 and 12 and is not corrected until the end. Observe that our algorithm ensures resynchronization at the end of the sequence ; this property is given for free and doesn't need to be based on a "reversibility" property of the VLC. Notice that although the symbol counter is not correctly estimated in the central part of the bit stream, symbol boundaries are correct. This is due to the so-called "resynchronization" property of VLCs. As a consequence, the estimated bit stream is correct in this area. This is evidenced by the central curve that displays the true symbol sequence and the estimated sequence, following the bit clock (each symbol value s is repeated l times if $\mathcal{L}(s) = l$). However, the desynchronization becomes obvious on the symbol clock axis (bottom curve). These curves illustrate the fact the a reasonable BER may nevertheless lead to a dramatic SER.

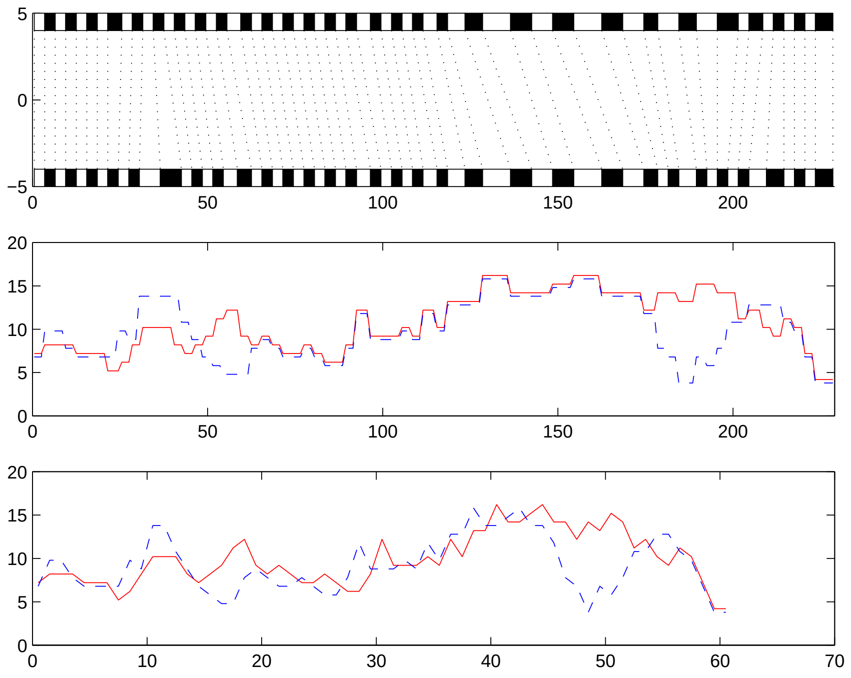


Figure 16: *Illustration of synchronization losses. The black and white patches on the top curve display symbol lengths, for a sequence of 60 symbols. The estimated sequence is on top, the true one below. The symbol correspondence is also represented. The two other curves show the difference between the estimated (dashed) and actual (solid) symbol values for each instant of the bit clock (center) and of the symbol clock (bottom).*

We have studied the resynchronization power of our iterative joint decoder by summing over n the difference between the true value of K_n and the estimated one. The result is divided by the number of bits in the sequence, which expresses the average desynchronization in symbols per bit. For the source model we considered (with high inter-symbol correlation), iterations are crucial for the resynchronization : at the fourth iteration, no desynchronizations were found for $E_b/N_0 > 1dB$, while symbol errors still remain.

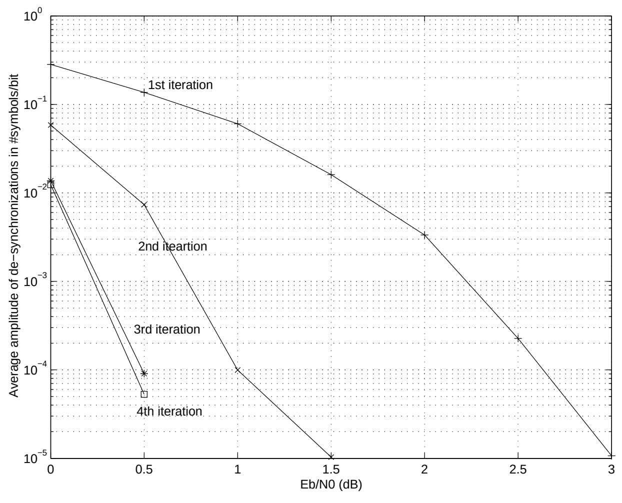


Figure 17: *Amplitude of desynchronizations at the different iterations of the joint source-channel turbo decoding. The results have been obtained for a Gauss-Markov source of 200 symbols quantized on 4 bits. They have been averaged over 500 AWGN channel realizations.*

Finally, let us mention that the effect of constraints on the numbers of symbols and bits appears mostly at the extremities of the bit stream, where the trellis becomes narrow. Nevertheless, even at low SNRs, the uncertainty on the value of K_n remains reasonably concentrated around its optimal value, even in the central part of the trellis. Fig. 18 displays the logarithm of the posterior distribution $\mathbb{P}(K_n|Y)$ for each value of n, assuming a channel noise level $E_b/N_0 = 0dB$. Similar curves appear also for white sources (although the beam is larger). This figure evidences that the complexity of estimation algorithms can be significantly reduced by pruning methods, which would not explore all nodes of the trellis. Preliminary results show that the complexity can be reduced by 50% to 90% without significant loss in BER.

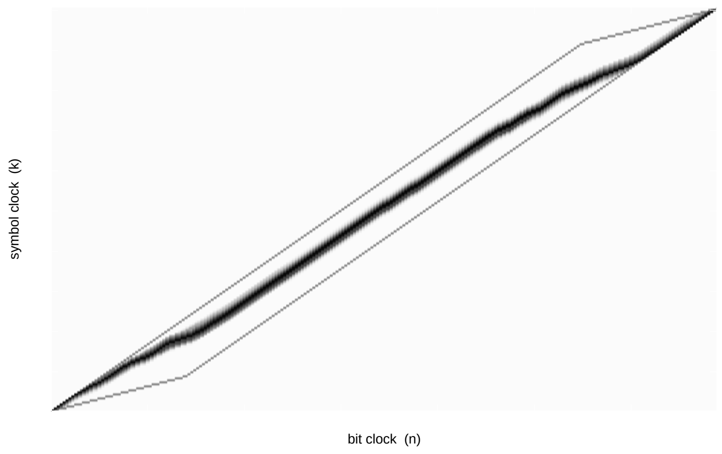


Figure 18: *Concentration of K_n on the trellis. The logarithm of $\mathbb{P}(K_n|Y)$ is displayed in gray scale, for a 200 symbol sample path of the Gauss-Markov source, and $E_b/N_0 = 0dB$.*

7 Conclusion

We have proposed a methodology for modelling a general coding chain composed of three elements : a Markov source, a variable length source coder and a channel coder. This model is studied in the formalism of Bayesian networks, from which estimation (i.e. decoding) algorithms derive immediately. The optimal joint decoding algorithms must be based on a product model gathering state representations of the three elements of the chain. This product model is too large to have any practical use, except in trivial cases. However, following properties evidenced in serial turbo-codes, joint decoding can be performed in an iterative procedure considering one factor of the product model at a time. This procedure usually requires the insertion of an interleaver between the components (or factors) of the model that will be processed separately, and is based on the exchange of soft information between the dedicated decoders. In the present case, one interleaver must be placed between the source coder and the channel coder, which brings some light technical difficulty since this interleaver must also be "variable length." But, surprisingly, the Markov source and the source coder need not be separated by another interleaver. Actually, it can be proved that a soft source decoding followed by a symbol stream estimation is an optimal strategy. This result is straightforward for constant length source codes, and it is quite surprising that it still holds for variable length source codes.

The scheme proposed in the present paper can be read as a turbo algorithm alternating the use of the Markov symbol source model and the channel coder model, which both introduce redundancy in the bit stream sent over the channel. The soft decoders for these extremal components communicate through the source decoder, which can be read as a translator of soft information from the bit clock to the symbol clock. This soft source decoder relies on two kinds of information : the residual intra codeword redundancy (which is quite low in the case of entropic coding), and mostly the length constraint for the bitstream. The latter ensures that the K symbols sequences produced by the source model do match the N bits sequences proposed by the channel decoder. The role of this constraint is crucial for variable length codes : noisy channels tend to "desynchronize" the bit clock and the channel clock. And errors in the estimation of codeword boundaries result in dramatic symbol error rates at the receiver. Reversible variable length codewords have been designed against this phenomenon. This reversibility property is useless for the algorithm we propose, since synchronization is ensured both at the beginning *and at the end* of the bit stream. Hence only the augmented internal redundancy of reversible codes is useful against desynchronizations. Nevertheless, the problem can be addressed directly, by inserting dummy symbols in the symbol stream, at some known positions, which serve as anchors for source decoding. This "soft synchronization idea" is currently being investigated, and has proved to augment considerably the autosynchronization power of the coding chain for very reasonable losses in information rate.

Finally, let us stress the parallel of variable length decoding of Markov sources and speech recognition. Symbols can be associated to words of a sentence, satisfying a Markov model, and variable length codewords can be compared to the variable number of acoustic segments in the pronunciation of a given word. The same problems of joint segmentation and word estimation have been addressed in the literature (see [24] and references therein), mainly with dynamic time warping (DTW) algorithms. The connection could be inspiring, in particular for pruning techniques, and for the availability of specialized chips implementing these algorithms.

References

[1] K. Sayood and J. C. Borkenhagen, "Use of residual redundancy in the design of joint source-channel coders," *IEEE Trans. on Communications*, vol. 39, pp. 838–846, June 1991.

[2] F. Alajaji, N. Phamdo, and T. Fuja, "Channel codes that exploit the residual redundancy in celp-encoded speech," *IEEE Trans. on Speech and Audio Processing*, vol. 4, no. 5, pp. 325–336, September 1996.

[3] N. Phamdo and N. Farvardin, "Optimal detection of discrete markov sources over discrete memoryless channels - applications to combines source-channel coding," *IEEE Trans. on Information Theory*, vol. 40, pp. 186–193, January 1994.

[4] K. Sayood et al., "A constrained joint source-channel coder design," *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 1584–1593, December 1994.

[5] K.P. Subbalakshmi and J. Vaisey, "Joint source-channel decoding of entropy coded markov sources over binary symmetric channels," in *Proc. International Conference on Communication, ICC*, March 1998, p. session 12.

[6] D. J. Miller and M. Park, "A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden markov models," *IEEE Trans. on Communications*, vol. 46, no. 2, pp. 222–231, February 1998.

[7] M. Park and D.J. Miller, "Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs," in *Proc. Conf. on Info. Sciences and Systems*, May 1998.

[8] A.H. Murad and T.E. Fuja, "Joint source-channel decoding of variable length encoded sources," in *Proc. Information Theory Workshop, ITW*, June 1998, pp. 94–95.

[9] N. Demir and K. Sayood, "Joint source-channel coding for variable length codes," in *Proc. IEEE Data Compression Conference, DCC*, March 1998, pp. 139–148.

[10] R. Bauer and J. Hagenauer, "Iterative source-channel decoding using reversible variable length codes," in *Proc. IEEE Data Compression Conference, DCC*, March 2000, pp. 93–102.

[11] R. Bauer and J. Hagenauer, "Turbo fec/vlc decoding and its application to text compression," in *Proc. Conference on Information Theory and Systems*, March 2000, pp. WA6.6–WA6.11.

[12] R. Bauer and J. Hagenauer, "Iterative source/channel decoding based on a trellis representation for variable length codes," in *Proc. Int. Symp. on Information Theory, ISIT*, June 2000, p. 238.

[13] R. Bauer and J. Hagenauer, "Symbol-by-symbol map decoding of variable length codes," in *Proc. 3rd ITG Conference on Source and Channel Coding*, January 2000, pp. 111–116.

[14] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. on Communications*, vol. 44, no. 10, October 1996.

[15] B.J. Frey and D.J.C MacKay, “A revolution: belief propagation in graphs with cycles,” in *Proc. of the Neural Inform. Processing Systems Conf.*, December 1997.

[16] R.J. McEliece, D.J.C. MacKay, and J.-F. Cheng, “Turbo decoding as an instance of pearl’s belief propagation algorithm,” *IEEE J. on Sel. Areas in Com.*, vol. 16, no. 2, pp. 140–152, February 1998.

[17] S.L. Lauritzen, “Graphical models,” Tech. Rep., Oxford Statistical Science Series 17, Oxford University Press, 1996.

[18] J. Pearl, “Fusion, propagation, and structuring in belief networks,” *Artificial Intelligence*, vol. 29, pp. 241–288, 1986.

[19] E. Fabre, “New fast smoothers for multiscale systems,” *IEEE Trans. on Signal Processing*, vol. 44, no. 8, pp. 1893–1911, August 1996.

[20] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. on Information Theory*, vol. 20, pp. 284–287, March 1974.

[21] A. Benveniste, B.C. Levy, E. Fabre, and P. Le Guernic, “A calculus of stochastic systems: specification, simulation, and hidden state estimation,” *Theoretical Computer Science*, , no. 152, pp. 171–217, 1995.

[22] N. Wiberg, “Codes and decoding on general graphs,” Tech. Rep., Ph.D Thesis, Dep. of Elec. Eng., Linköping Univ., 1996.

[23] R.J. McEliece and S.M. Aji, “The generalized distributive law,” *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 325–343, March 2000.

[24] M. Ostendorf, V. Digilakis, and O. A. Kimball, “From hmm to segment models: A unified view of stochastic modeling for speech recognition,” *IEEE Trans. on Speech and Audio Processing*, vol. 4, no. 5, pp. 360–378, September 1996.

[25] D.J.C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. on Information Theory*, January 1999.

[26] F.R. Kschischang and B.J. Frey, “Iterative decoding of compound codes by probability propagation in graphical models,” *IEEE J. on Sel. Areas in Com.*, vol. 16, no. 2, pp. 219–230, February 1998.

[27] Y. Takishima, M. Wada, and H. Murakami, “Reversible variable length codes,” *IEEE Trans. on Communications*, vol. 43, no. 4, pp. 158–162, April 1995.

[28] L. Guivarch, J.C. Carlach, and P. Siohan, “Joint source-channel soft decoding of Huffman codes with turbo codes,” in *Proc. IEEE Data Compression Conference, DCC*, March 2000, pp. 82–92.