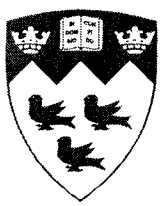
NOTE TO USERS

An Enhanced Joint Source-Channel Decoder

Karim Ali



Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

May 2005

A thesis submitted to McGill University in partial fulfillment for the degree of Master of Engineering.

2005/05/12

Your file Votre référence
ISBN: 0-494-12577-2
Our file Notre référence
ISBN: 0-494-12577-2

Sommaire

Il a été démontré qu'un encodage et décodage tandem atteint des taux d'erreurs arbitrairement bas étant donné une longeur de bloc suffisamment élevée. Cependant, lorsque appliquée à des systèmes pratiques qui sont intrinsèquement limités en terme de compléxité et donc en terme de longueur de bloc, la stratégie tandem peut être largement sous-optimale. En effet, un décodage tandem ignore deux types d'information: la mémoire de la source ansi que la redondance résiduelle de l'encodeur de source. De plus, les décodeurs de source conventionels, dans une stratégie de décodage tandem, sont conçus pour accomplir l'application inverse de l'encodeur de source et peuvent donc causer une détérioration importante de la performance dans le cas où des erreurs seraient encore présentes à leur entrée. La conception d'un décodage joint source-canal, qui prendrait en considération les deux types de redondances additionelles — c'est à dire, la mémoire de la source et la redondance résiduelle de l'encodeur de source — est une possibilité viable qui génère nécessairement des gains.

Dans ce contexte, nous proposons un nouvel algorithme itératif de décodage joint source-canal. Cet algorithme est dérivé d'une représentation par réseaux Bayesiens de la chaîne de codage et prend en compte trois types d'information: la mémoire de la source, la redondance résiduelle de l'encodeur de source ainsi que la redondance amenée par l'encodeur de canal. Plus précisemment, nous modifions un algorithme existant en dérivant une nouvelle représentation équivalente par réseaux Bayesiens de la chaîne de codage. De plus, nous proposons une nouvelle méthode, entièrement consistente par rapport au cadre des réseaux Bayesiens, pour accomplir les itérations. Lorsque comparé avec l'algorithme existant, notre algorithme montre non seulement des gains substantiels mais encore une importante réduction de complexité. Enfin, nous exposons quelques améliorations additionelles qui peuvent être apportées à notre algorithme. Ces dernières incluent des méthodes de réduction de complexité supplémentaires qui dans un cas, viennent sans coût en terme de performance et dans un autre cas, avec un moindre coût en performance.

Abstract

Tandem coding and decoding has been demonstrated to yield arbitrarily low error rates provided a sufficiently large block length is used. When applied to practical systems that are inherently limited to a finite complexity and therefore to finite block lengths, such a strategy may be largely suboptimal. Indeed, a tandem decoding strategy ignores two types of information: the source memory and the residual redundancy of the source coder. Moreover, conventional source decoders, within a tandem decoding strategy, are designed to perform the inverse operation of the source coder and may severely decrease performance if errors are still present at their input. One viable alternative, that has been demonstrated to yield gains, is the design of a joint source-channel decoding scheme that would take the additional sources of redundancies — the source memory and the source coder's residual redundancy — into account.

In this context, we propose a novel, iterative joint source-channel decoding algorithm. The proposed scheme is derived from a Bayesian network representation of the coding chain and incorporates three types of information: the source memory; the residual redundancy of the source coder; and finally the redundancy introduced by the channel coder. Specifically, we modify an existing algorithm by first deriving a new, equivalent Bayesian network representation of the coding chain. Next, we derive a fully consistent methodology, within the framework of Bayesian networks, for performing the iterations. The proposed algorithm is shown to yield significant gains along with a drastic reduction in computational complexity when compared with the existing one. Finally, we outline additional possible improvements on the proposed algorithm. They include methods for further reductions in computational complexity at no cost in performance in one case, and at a slight cost in performance in the other.

Acknowledgments

My gratitude goes to Professor Fabrice Labeau for his guidance and time. I would also like give thanks to the *Fond de recherche sur la nature et les technologies* for their financial support. I further thank Ms. Tania Leppert, Mr. Nikolaos Gryspolakis, Mr. Faker Moutamri and Mis. Suqun Fan for our fruitful discussions. Finally, I would like to offer special thanks for the individuals that have helped me with the various programming difficulties that I encountered: Mr. Martin Cudnoch, Mr. Eric Bertrand, Mr. Rui Ma and Dr. Anton Vinokurov.

Contents

1 Joint Source-Channel Coding and Decoding 4
1.1 Coding Theorems . . . 4
1.1.1 Source coding . . . 4
1.1.2 Channel coding . . . 7
1.2 The Joint Coding Problem . . . 10
1.2.1 Source-channel theorem . . . 10
1.2.2 A heuristic motivation for joint coding . . . 11
1.3 The Joint Decoding Problem . . . 13
1.3.1 Joint decoding as a separate problem . . . 13
1.3.2 Optimal joint decoding solution . . . 14
1.3.3 Need for suboptimal joint decoding schemes . . . 16

2 Bayesian Networks and Belief Propagation 18
2.1 Defining a Bayesian Network . . . 18
2.2 Emergent Properties . . . 21
2.2.1 Computational savings . . . 22
2.2.2 Conditional independence relations . . . 22
2.3 Pearl's Belief Propagation . . . 25
2.3.1 Discrete Polytrees . . . 25
2.3.2 Discrete Trees . . . 31
2.3.3 Continuous observation on a discrete node . . . 32
2.4 More on Belief Propagation . . . 33
2.4.1 Organized strategies . . . 33
2.4.2 Belief Propagation and the inference problem . . . 33
2.4.3 Graphs with loops . . . 35

3 Joint Source-Channel Decoding via Bayesian Networks **36**
3.1 The Joint Decoding Problem . . . 37
3.2 Deriving the Bayesian Network Representation of the Coding Chain . . . 37
3.2.1 Preliminaries . . . 37
3.2.2 The Markov source and source coder . . . 38
3.2.3 The channel coder . . . 41
3.2.4 The entire coding chain . . . 42
3.3 Joint Decoder . . . 43
3.3.1 Incorporating knowledge of the received data stream . . . 44
3.3.2 Incorporating knowledge of the transmitted symbol sequence length . . . 45
3.3.3 Applying Belief Propagation on the available graph . . . 45
3.3.4 Turbo joint decoding scheme . . . 47

4 An Enhanced Joint Source Channel Decoder: Theory and Results **49**
4.1 Proposed Algorithm . . . 49
4.1.1 An equivalent Bayesian network representation of the coding chain . . . 50
4.1.2 Towards a fully consistent solution for turbo joint decoding . . . 54
4.2 Theoretical Discussion on the Proposed Algorithm . . . 57
4.2.1 Improved Convergence Properties . . . 57
4.2.2 Computational Complexity Reduction . . . 60
4.3 Comparative Study: Results and Discussion . . . 64
4.3.1 Blocks of 50 symbols . . . 64
4.3.2 Blocks of 200 symbols . . . 69
4.4 In Retrospect . . . 71

5 Other Improvements and Preliminary Results **72**
5.1 Further Reductions in Computational Complexity . . . 72
5.2 Bit Simplification . . . 74
5.3 The Effects of Inexact Knowledge of $P(s_n|s_{n-1})$ . . . 76
5.4 The Effects of the Interleaver and Recursive Convolutional Code . . . 77
5.5 Anti-Causal Graph . . . 78

6 Conclusion **79**

A Computational Complexity Analysis **82**
A.1 Black Box Implementation of Belief Propagation . . . 82
A.2 Efficient Implementation of Belief Propgation for Sparse Matrices . . . 85

A.3 The Sparse Matrices in the Bayesian network representation of the coding chain . 87
A.3.1 Without knowledge of N 87
A.3.2 With knowledge of N 88

B Additional Results **91**

References **94**

Notational convention

$\|\mathcal{S}\|$	Cardinality of a set.
$\mathcal{A} \times \mathcal{B}$	Cartesian product of the set $\mathcal{A}$ with the set $\mathcal{B}$.
X	Random variable X.
$\mathcal{X}$	Set on which random variable X assumes values.
x	Generic realization of the random variable X.
$P(.)$	Probability measure on an implicitly assumed probability space.
$P(x)$	Shorthand notation for $P(X = x)$. Represents probability of the event $X = x$.
$P(x\|y)$	Shorthand notation for $P(X = x\|Y = y)$. Represents conditional probability of the event $X = x$ given event $Y = y$.
$P(x, y)$	Shorthand notation for $P(X = x, Y = y)$. Represents joint probability of $X = x$ and $Y = y$.
$P(x, y) = P(x\|y)p(y)$	Equality is understood to hold over all possible events $X = x$ and $Y = y$.
$p(x)$	Probability mass function for random variable X.
$x^{(i)}$	Used when needed to distinguish the different possible values of x.

Preamble

Systems that employ separate encodings, namely source coding followed by channel coding, are ubiquitous. Indeed, separate encodings simplify the coding operation, breaking it into two dual and separately defined tasks: the removal of natural redundancy present in the data on one hand, and the addition of artificial redundancy for error resilience against the channel on the other. On the decoding end, the same phenomenon holds as separation entails the data to be decoded sequentially, first using the appropriate channel decoder and next the source decoder, designed to perform the inverse operation of the source coder. A separate encoding strategy possesses more tangible advantages such as inter-operability, meaning the ability to easily adapt to the transmission of data obtained from different sources, simply by changing the source encoder. More importantly, it has been demonstrated that arbitrarily low error rates may be achieved with separate encodings and decodings, under the proviso of choosing sufficiently large block lengths.

The situation in practice is somewhat different since the proviso of larger block lengths immediately translates to higher complexities that are simply unaffordable with constraints such as delay. For this reason the bottleneck of coding theory is embodied by the problem of obtaining the most performance, in terms of error rate, for a given complexity. Some researchers have suggested and demonstrated the possibility of a joint coding strategy, namely the design of a code that would take both the characteristics of the source and those of the channel into account, outperforming the separate encoding strategy for a given complexity. Such systems, that utilize joint coding, naturally use joint decoding on the receiver end, in order to capitalize on potential gains.

However, it is also possible to consider the possibility of using joint decoding for systems that employ separate encodings. The premise is that practical systems that utilize separate encodings must necessarily use finite complexity and therefore finite block lengths for the source coder and the channel coder. This in turn implies that the data at the output of the source coder possesses additional redudancies — the residual redundancy of the source coder and the source memory (inter-symbol correlation). Whereas these redundancies are necessarily present in the received data stream, separate decodings will simply ignore them. One can therefore consider designing

a joint decoder that would incorporate the two former sources of *natural* redundancy along with that *artificial* redundancy introduced by the channel coder; a possibility mentioned as early as Shannon's seminal paper [2]. Such a design strategy is motivated further by the fact that optimal source coders of the variable length code variety have corresponding source decoders that are extremely sensitive to noise: the lack of set symbol boundaries resulting in a vulnerability to synchronization errors. Joint decoders, therefore, are a viable alternative that will necessarily imply performance gains.

Murad et al. [15] developed a generic solution to the joint decoding problem by deriving the product finite state machine model of the source, the source coder and the channel coder. Various algorithms such as Hard Viterbi, Soft Viterbi and BCJR (Kalman smoothing) are then readily applicable yielding the *optimal* solution with respect to the algorithms' criteria. Unfortunately, this solution has intractable complexity. This phenomenon leads to the need for less complex and therefore sub-optimal joint decoders. In this context, the authors in [17]-[18] provided a sub-optimal joint decoding solution under the additional assumption of a memoryless source. Specifically, their proposed algorithm uses the principle of turbo-decoding and alternates the use of a soft source decoder with a soft channel decoder. This approach was recently extended in [20] to include sources with memory. The algorithm, which also relies on the principles of turbo-decoding and was derived in the context of Bayesian networks, has the advantage of isolating the constituent components and therefore has limited complexity.

Contribution and Organization

In this text we present an enhanced sub-optimal joint decoder that is largely inspired from the developments of [20], and incorporates three types of redundancies: the source memory, the source coder's residual redundancy and the artificial redundancy of the channel coder. In particular, we first derive a new, equivalent Bayesian network representation of the coding chain. Next, we derive a fully consistent methodology, within the framework of Bayesian networks, for effecting the iterations. The proposed algorithm is shown to yield significant gains along with a drastic reduction in computational complexity when compared with the existing one [20].

This text is organized as follows. In Chapter 1, we attempt to frame the joint decoding problem within the larger context of coding. Specifically, we introduce and define the notions of source and channel coding. Next we develop on the separation principle, expose the possible advantages of joint coding/decoding and define the problem of joint decoding separately from joint coding. Finally, we examine the optimal joint decoding solution and explore the need for sub-optimal joint decoders.

In Chapter 2, Bayesian networks are seen to provide a graphical framework for the analysis of statistical problems. Belief Propagation is derived from first principles and is shown to be an efficient, graphically based solution to the inference problem that may be used in the context of decoding.

Chapter 3 is based on the developments of [20] and shows how the joint decoding problem may be approached and analyzed in the context of Bayesian Network. Specifically, we consider the derivation of the Bayesian network corresponding to the entire coding operation and demonstrate how the resulting graph may be used by the receiver — which has the additional knowledge of the received data stream and possibly the length of the received symbol sequence length — in a sub-optimal yet robust joint iterative decoding scheme.

In Chapter 4, we present our proposed algorithm. We first derive in detail the algorithm that is based on one hand on an equivalent Bayesian network representation of the coding chain and on the other, on a different approach with respect to the iterations. The theoretical analysis indicates both better convergence properties as well as a reduction in computational complexity when compared with [20]. These expected results are, next, substantiated by experimental computer simulations.

Finally, Chapter 5 depicts in detail the possibilities of additional improvements, both in terms of computational complexity reduction and performance amelioration.

Chapter 1

Joint Source-Channel Coding and Decoding

This chapter aims at elucidating the various considerations that have led researchers to seriously consider the problem of joint coding and decoding. This point merits particular attention since it is commonly accepted that the transmission problem may be broken into two separate tasks: source coding and channel coding. Indeed the source-channel theorem dictates that if infinite complexity is allowed, there is no loss in optimality in such a strategy. However researchers have recently pointed out that this optimality is only *asymptotic* and does not necessarily hold for practical systems. We attempt to clarify this point in order to better frame the problem of joint coding and joint decoding that we are interested in. To this end, we begin this chapter with a brief and concise review of the coding theorems. Second, the joint coding problem is examined. Specifically, the source-channel theorem is stated and a brief discussion with respect to its implications ensues; next we attempt to give a heuristic motivation for joint coding along with some examples of attempts at joint coding. Finally, we introduce the problem of joint decoding that, as will be seen, can be considered as separately defined from joint coding. In this setting, we present the optimal joint decoding solution and explore the need for suboptimal joint decoders.

1.1 Coding Theorems

1.1.1 Source coding

The ultimate objective of source coding is to achieve data compression[1]. This is motivated in practice by the need to store data in its most efficient form, removing all superfluous or unwanted

[1]The terms data compression and source coding are to be used interchangeably.

content. There are two types of source coding: *lossy* coding and *lossless* coding. Lossy coding assumes that the original data is not to be recovered in its entirety. Such a situation occurs for example when a continuous-time signal is sampled and quantized in order to be stored in a digital form. There is no fundamental limit on lossy source coding[2] since we may choose to get rid of as much information from the data as desired. Lossless coding on the other hand assumes full recovery of the original data. A Fourier series decomposition of a periodic signal satisfying Dirichlet's conditions is in principle lossless coding since the entire signal can be reconstructed from its real Fourier coefficients. However this scheme is utterly impractical since an infinite amount of storage space on a general purpose device is required to store just one real sample. This is not to say that Fourier decomposition is not a valid data compression scheme: the original data is now represented in a far more compact manner, yet the method does not provide practically implementable solutions.

Source coding theorem

The difficulty stated above arises more generally when attempting lossless source coding on discrete data with either uncountable alphabets or countably infinite alphabets. Most practical systems will implement lossy coding as a first stage — sampling, quantization — before considering the problem of lossless coding on the now finite-alphabet discrete data. For this reason, we consider the case of lossless source coding applied to discrete finite-alphabet data and refer to source coding in this context. The guiding principle of data compression in this case is to assign short descriptions to the most frequent outcomes of the data and necessarily longer descriptions to the less frequent outcomes. It is convenient to consider data as a *stochastic process* so that we may refer to the likelihood of a particular symbol or sequence of symbols. A source code can then be a mapping from each time sample of the stochastic process (symbol) to a set of finite length strings (codewords). With such a definition in mind, each source code is nothing more than a particular representation of the data, according to the chosen codewords. The smaller the average length of the codewords, the more compactly we have represented our data and the better the compression. We define a source code and then state the source coding theorem.

Definition 1.1. A *source code*, S, for a random variable X taking values on a discrete set $\mathcal{X}$ with probability mass function $p(.)$, is an injective mapping from $\mathcal{X}$ to D^*, the power set of the alphabet D. Let $S(x)$ denote the codeword corresponding to x, an instance of X and let $l(x)$ denote the length of $S(x)$. The *expected length*, L, of the source code is given by,

$$L = \sum_{x \in \mathcal{X}} p(x) l(x) \tag{1.1}$$

[2]Rate-distortion theory in fact provides fundamental limits in terms of rate-distortion functions

The definition of a source code given above rephrases mathematically the elements of the previous paragraph. Specifically, it states that a source code is nothing more than a representation of the data through finite length strings obtained by concatenating elements of D, which in most applications is $\{0,1\}$. One notable difference: given a stochastic process $\{X_i\}$, the definition of source code above is not restrictive to a mapping from symbols X_i to D^* but allows as well mappings from super-symbols $\{X_{i+1}, X_{i+2}, \ldots, X_{i+n}\}$ to D^*. In words, it is possible to assign finite length strings to a concatenation of n data symbols. By so doing, we are achieving compression on sequences of symbols instead of symbols alone and hence, the inter-symbol correlation can be removed. The quantity n is referred to as the *block length* of the code. The expected length of a source code is the statistical average of the codeword lengths and provides a measure for the code's performance or efficiency: a source code with a small expected length will likely require less storage space, to store the same data, than a source code with a larger expected length. The source coding theorem, stated below, defines the limit to data compression.

Theorem 1.1. [1] Let $\{X_i\}$ be a discrete stationary ergodic stochastic process. Let L_n^{min} be the minimum expected codeword length per symbol over all possible source codes of block length n. Then,

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \leq L_n^{min} < \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n} \tag{1.2}$$

We use $H(.)$ to denote the *entropy* of a random variable and we will refer to the quantity,

$$\lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) \tag{1.3}$$

as the entropy rate[3] of the process and denote it as $H(\mathfrak{X})$. The theorem contains several points of interest. First, at any given block length, the best possible source code will have an expected length obeying equation 1.2. Second, as is implied by the first statement, there is no source code that will represent data in such a way that the expected length per symbol is smaller than the data's entropy rate. Therefore the entropy rate of the data is the fundamental limit to data compression. In view of the fact that entropy represents the uncertainty of a random variable or more precisely, its *true randomness* and information without which it is irrecoverable, it is indeed intuitively meaningful that the limit to source coding is related to the data's entropy. Compressing the data down to its entropy entails that all redundancy is removed and hence a representation with i.i.d equiprobable elements of D is necessarily obtained[4]. Thirdly, we may also deduce through a limiting argument, that with larger block lengths, one can find a sequence of source codes with an expected length per symbol that asymptotically approaches entropy. Restated another way,

[3] the entropy rate of a stationary ergodic stochastic process is always well defined.

[4] any other distribution implies that redundant information is still present.

it implies that it is possible to compress data arbitrarily close to to its entropy rate, if we use a sufficiently large block length. The theorem was first proven by Shannon [2] under the additional assumption that the process is i.i.d and the proof actually provided guidelines for the construction of a code satisfying inequality 1.2. It was shown that the assignment $l(x) = \lceil \log(\frac{1}{P_X(x)}) \rceil$ allows the construction of a source code that compresses the data within one bit of entropy. Applying the same method on larger super-symbols, one can get arbitrarily close to entropy.

On a final note, source codes are generally split into two categories in the literature: constant length codes (CLC's) and variable length codes[5] (VLC's). CLC's assume that a fixed codeword length is to be used for all data symbols or super-symbols while VLC's relax that assumption and allow variable codeword length. It is important to mention that source codes that obey equation 1.2 are called *optimal* since they provide us with the best possible compression for a given block length. For sources with unequal symbol probabilities, it should be clear that most optimal codes are of the VLC variety[6]. One such class of codes can be obtained through the well established method called the Huffman algorithm. Huffman codes are particularly interesting because they are easily implementable if one has access to the statistics of the data, but more importantly, they are optimal. Other algorithms such as the Lempel-Ziv algorithm, run-length limited coding and Tunstall coding also yield optimal source codes.

1.1.2 Channel coding

The fundamental goal of channel coding is to protect data against corruption during a wireless or wireline transmission. Corruption occurs for various reasons during a transmission, reasons that are outside the control of the sender and the receiver. Thermal noise, destructive interference caused by echoes or other transmissions, fading and data collision all contribute to errors in the received data stream. It is impossible to directly eliminate the causes of corruption at their source and reduce the likelihood of errors in such a manner. What is possible however is to mitigate the effects as much as possible through the insertion of redundant information that will help protect the data stream. If redundant information is sent along with the original data and the receiver is aware of the scheme, it becomes intuitively conceivable that some errors may be recovered at the receiving end. Here, we define the structure of communication systems with a channel code and subsequently state the channel coding theorem.

Consider figure 1.1 below of a general communication system with a channel code. There are three essential components — the channel encoder, the communication channel and the channel

[5] equivalently, fixed rate codes and variable rate codes.

[6] for sources with equiprobable or quasi-equiprobable symbols, CLC's should perform as well or nearly as well.

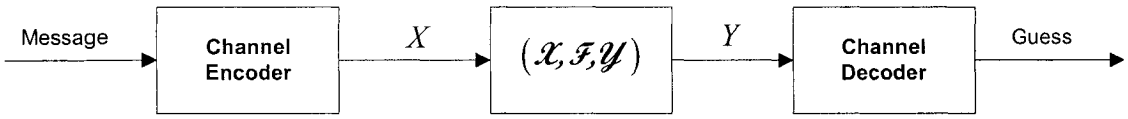


Fig. 1.1 General communication system with channel code.

decoder — which we now define.

Definition 1.2. A general communication channel consists of a set $\mathcal{X}$ called the input alphabet, a set $\mathcal{Y}$ called the output alphabet as well as a set $\mathcal{F}$ of conditional probability measures relating $\mathcal{X}$ to $\mathcal{Y}$. At every time instant, an element of $\mathcal{X}$ is selected for transmission and subsequently mapped to an element of $\mathcal{Y}$ according to the appropriate[7] element of $\mathcal{F}$. We denote a communication channel by $(\mathcal{X}, \mathcal{F}, \mathcal{Y})$. With this broad definition in mind, different classes of channels may be obtained by considering the various restrictions that can be placed on the set $\mathcal{X}$, the set $\mathcal{Y}$, the set $\mathcal{F}$ of probability measures modeling the effects of corruption, as well as on the nature of the time index.

The function of the channel encoder is to add redundancy to the data. This is usually done by representing each data symbol with more information, the new information being deterministically related to the original one. The function of the channel decoder is to make the best possible guess about the source symbols based on the received data. We assume that the data to be transmitted is drawn from the index set $\{1, 2, \ldots, M\}$ for generality.

Definition 1.3. An (M, n) channel encoder, with $n > \lceil \frac{\log(M)}{\log|\mathcal{X}|} \rceil$, for the channel $(\mathcal{X}, \mathcal{F}, \mathcal{Y})$ is an injective mapping from the index set $\{1, 2, \ldots, M\}$ to the set $\mathcal{X}^n$,

$$X^n : \{1, 2, ...M\} \rightarrow \mathcal{X}^n \tag{1.4}$$

yielding codewords $X^n(1), X^n(2), \ldots, X^n(M)$. The set of codewords is called the codebook and the quantity n is once again referred to as the block length of the code. The corresponding channel decoder is a mapping from $\mathcal{Y}^n$ to $\{1, 2, ...M\}$

$$g : \mathcal{Y}^n \rightarrow \{1, 2, ...M\} \tag{1.5}$$

a deterministic rule assigning a guess to each possible received vector.

There does exist a duality between the definition of source coding and that of channel coding. They are both injective mappings and where one tries to remove redundancy, the other adds.

[7] Depending on the current time, the current input and possibly, past inputs and outputs.

Intuitively, assuming all alphabets to be binary, the definition of a channel code above states that we must map every symbol in our data from the set $\{0,1\}^{\lceil log_2(M) \rceil}$ to a distinct codeword in the set $\{0,1\}^n$. Since $n > \lceil \log_2(M) \rceil$, we are now using more bits per symbol and hence we are adding redundant information. A common strategy is to try to distinguish the codewords in $\{0,1\}^n$ as much as possible by maximizing the distance between the codewords. The Hamming distance which returns the total number of coordinates (bit positions) in which the codewords are different is especially useful. The decoding strategy is then to match the received codeword with the closest known one in terms of Hamming distance: the hard input Viterbi algorithm essentially implements that process. Channel codes that have greater distances between their codewords are clearly less susceptible to error and in fact, the minimum Hamming distance between two codewords provides an important measure that sets the lower bound on error rate. In the remainder of this text, we will assume all alphabets to be binary. The quantity $R = \frac{\log_2(M)}{n}$ is called the rate of the code and we equivalently denote an (M,n) code by $(\lceil 2^{nR} \rceil, n)$ or simply $(2^{nR}, n)$.

Channel coding theorem

The channel coding theorem essentially justifies the use of channel coding for the purposes of error correction. It states that it is possible to reduce the probability of error arbitrarily close to zero by choosing an appropriate channel code with sufficiently large block length.

Theorem 1.2. [1] Let R represent the rate measured in bits per channel use that we wish to transmit at. For every rate $R < C$, there exists a sequence of $(2^{nR}, n)$ channel codes with a maximum probability of error tending to zero as n increases to infinity. Conversely, any sequence of $(2^{nR}, n)$ channel codes with a maximum probability of error tending to zero as n increases to infinity must have $R \leq C$.

The quantity C, called the channel capacity, depends on the class of channel considered. Shannon [2] initially proved his theorem for the case of the Discrete Memoryless Channel[8]. The theorem was indeed surprising for researchers had believed that the uncontrollable effects of corruption necessarily meant that an error floor existed for any rate of transmission. Unlike Theorem 1.1, the channel coding theorem does not provide as useful guidelines for the construction of good channel codes since the proof relies on random codes. Such codes may be used and do provide good results, however they are very difficult to decode and entail a high degree of complexity. Dobrushin [3] proved the theorem for the class of *information stable*[9] channels and

[8]Such a channel assumes countably finite $\mathcal{X}$ and $\mathcal{Y}$. In terms of the restrictions on $\mathcal{F}$, the set is time invariant, its elements are conditionally dependent on the current element of $\mathcal{X}$ alone, and the current output is statistically independent of future inputs.

[9]Those channels can be roughly described as having the property that the input that maximizes mutual information and its corresponding output behave ergodically.

finally Verdú [4] proved the theorem for arbitrary non-feedback channels and established the most general formula for capacity.

1.2 The Joint Coding Problem

In light of the previous section, we can now reconsider the problem of sending data obtained from a discrete finite alphabet source. For example, suppose we want to transmit English text over an erasure channel[10]. We could design a *joint code* that can consider the characteristics of the source and at the same time those of the channel so as to find an optimal way of mapping the sequence of letters directly to the input of the channel. Or we could use a *two-stage* method before sending the information: first compress the text as efficiently as possible and subsequently use an appropriate channel code, designed for the channel, to add redundancy. The question of which of these two methods will imply the best performance is the topic of discussion.

1.2.1 Source-channel theorem

It turns out that it is indeed possible to combine the results of Theorem 1.1 and Theorem 1.2 and express the condition, under which it is possible to transmit reliably, in terms of the characteristics of the source. The source-channel theorem provides such a statement. It states that a sufficient and necessary condition for transmission with arbitrarily low error rate is that the entropy rate of the data be strictly smaller than channel capacity.

Theorem 1.3. [1] **Source-channel theorem:** A stochastic process $\{U_i\}$ with entropy rate $H(\mathfrak{U})$ cannot be sent reliably[11] over a channel if $H(\mathfrak{U}) > C$. Conversely, if the process is stationary and ergodic, then the source can be transmitted reliably if $H(\mathfrak{U}) < C$.

Shannon originally proved the theorem for the case of the discrete memoryless channel and the process $\{U_i\}$ was assumed to have finite alphabet. As is the case with the channel coding theorem, the source-channel theorem was later extended to include larger classes of channels. The part of interest to our discussion is the proof of the converse in which the aforementioned two-stage method is used. Specifically, it is shown that an arbitrarily low error rate can be reached with the two-stage method provided a sufficiently large block length is used for both codes. And because of the direct part of the theorem, it follows that either reliable transmission is possible with separate source-channel coding or it is not possible at all. The interpretation of the theorem was that one can therefore transmit data in a two-stage method with no loss in optimality. This had

[10] Such a channel produces no error when the data is received however it does have a certain probability that the transmitted symbol is lost.

[11] The term reliably in the statement is understood to mean that an arbitrarily small error rate can be reached provided a sufficiently large block length is used.

tremendous practical implications as it meant that we can consider the design of a communication system as a combination of two parts: source coding and channel coding as is shown in figure 1.2. Among other things, the separation principle entailed that when designing the source code, only

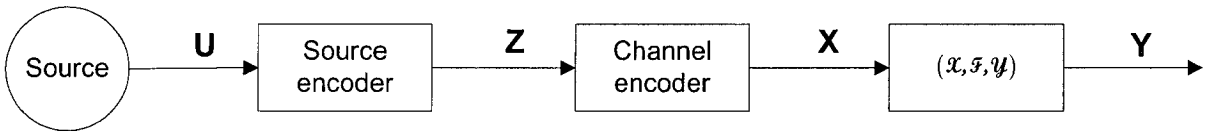


Fig. 1.2 Separation principle

the characteristics of the data need be considered and similarly when designing the channel code, only the characteristics of the channel are taken into account. The task of the source coder is therefore to remove as much natural redundancy from the data as possible and ideally present an input of i.i.d equiprobable bits to the channel coder. The latter's task is the reinsertion of artificial redundancy for error resilience against the channel's corruption. Hence the design strategy for a communication scheme is expressed in terms of two dual, yet well defined tasks. On the receiver end, the same phenomenon holds as we are able to decode the data sequentially first using the corresponding channel decoder and second the source decoder. The task of the channel decoder is to estimate the received sequence based on the channel coding scheme alone, whereas the source decoder, referring to definition 1.1, is the corresponding inverse[12] mapping.

1.2.2 A heuristic motivation for joint coding

Recently, however, the separation principle has been put to question. Indeed, the source-channel theorem's interpretation is very different in terms of its implications when compared with the source coding theorem or the channel coding theorem. The source coding theorem establishes the limit to compression; the channel coding theorem establishes the possibility of arbitrary error rate; the source-channel theorem is special in that it offers a *strategy* of design. The point of contestation is that the separation principle is only shown to be optimal as the block length of both codes increases to infinity. In other words, no statement is made concerning the case of finite complexity. Under such a constraint, it is not clear that the strategy of separation remains optimal: it is indeed possible that a joint coding strategy would generate a better performance. This point of contestation is motivated further by the fact that all practical systems are fundamentally limited to a finite complexity. More importantly, practical systems have severe limitations such as delay that further limit the complexity of the coding and decoding schemes.

[12] the latter is guaranteed to exist because of the definition of the source coder as an injective mapping.

Although a rigorous proof, concerning the sub-optimality of the two-stage method under finite complexity, has yet to be put forth, some works have laid strong theoretical foundations. Massey [5], who was amongst the first researchers to consider joint coding, showed that for a distortionless transmission across a binary symmetric channel, a significant reduction in complexity with equivalent performance to separate coding can be achieved using a joint source-channel coder. This, under the premise that linear (block) source and channel codes are used. On a more general note, the separation principle has been shown to break down for some examples of multiuser channels [1], and even some examples of single-user information stable channels [6].

Many other arguments exist that demonstrate the possibility of joint coding outperforming separate encoding under finite complexity. If we reconsider the example of sending English text across an erasure channel, we can note that English text, like all languages, has a significant amount of natural redundancy due to its grammatical structure, syntax and morphology. For this reason, if we send the English text directly over the channel, we can lose up to half the letters and yet still be able to decode the text. It does seem in this particular case that the natural redundancy of the data is well adapted to the channel and it is perhaps better to leave it intact for the purposes of error resilience against the channel. More generally, there is perhaps an advantage in having channel codes designed according to the characteristics of the source. It is clear that when a finite complexity source coder is used, the data presented to the channel coder is not perfectly i.i.d and equiprobable. If a block length of 1 is used in a binary compression scheme — a commonly used one — equation 1.2 tells us that there can be as much as 1 redundant bit per symbol left. The remaining natural redundancy, termed *residual redundancy* can of course be used by the channel coder. In addition, a block length 1 source coder leaves all source memory (inter-symbol correlation) intact and again the same argument applies.

There are some examples in the literature of such attempts. In [7], Cox et al. develop a method of passing important source information such as the statistics of the data, termed source significance information (SSI), to the channel coder. In particular, the SSI is used by the channel coder for both static and dynamic unequal error protection. Significant gains were obtained when compared to the separate counterpart of equal error protection. His approach was termed "source-controlled channel coding". In [8], Sayood presented a technique for providing error protection without the additional overhead of channel coding. The original premise was that imperfect source coding, due to lack of knowledge of the exact source statistics or due to complexity limitations, necessarily means that residual redundancy is present at the output of the source coder. He essentially provided a method of utilizing this redundancy much the same way that channel code redundancy is used. His technique showed substantial gains for image transmission over a discrete memoryless

channel with the standard DPCM source coding scheme. In essence, Sayood implemented the image transmission version of the English text example discussed earlier. His approach was later extended in [9] to the more widely used DPCM/convolutional coder combinations. In [10], Alajaji and Fuga considered the problem of designing channel codes that exploit the residual redundancy in CELP-encoded speech. His work focused on the fact the line spectral parameters (LSP's) of the CELP scheme contain a great amount of redundant information. Specifically, as many as one-third of the LSP bits in every frame of speech are redundant. He considered the design of adapted forward error control (FEC) codes as well as block codes and convolutional codes. Once again, significant gains were obtained under the widely utilized model of the Additive White Gaussian Noise (AWGN) channel. It is not surprising that all these works considered working on data such as speech, audio and image since they all contain a great deal of natural redundancy.

1.3 The Joint Decoding Problem

As previously mentioned, the separation principle entailed that on the receiver end, data can be decoded sequentially via the channel decoder first and the source decoder next. It is clear that when a joint coding strategy is used, one must also use a corresponding joint decoding strategy in order to capitalize on the possible gains of the former. In the case that a channel coder is designed to take into account the residual redundancy of the source coder, the joint source-channel decoder should in its turn rely on both sources of redundancy in its operations: the residual redundancy of the source coder and the artificial redundancy of the channel coder. All of the previously mentioned works on joint coding developed not only joint coding schemes, but also corresponding joint decoding schemes.

1.3.1 Joint decoding as a separate problem

The previous point notwithstanding, it is possible to consider the joint decoding problem as a separately defined one. Specifically, it is reasonable that joint decoding may be implemented on systems that utilize separate encodings (or tandem encoding) as is shown in figure 1.3. As we

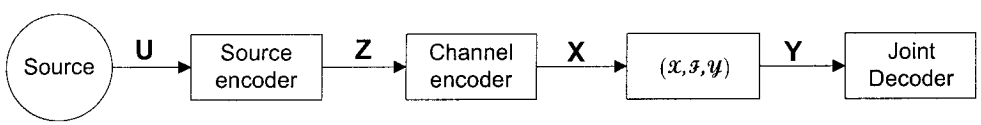


Fig. 1.3 Joint decoding for separately encoded systems.

have seen, when finite complexity separate encodings are used, residual redundancy is necessarily present at the input of the channel coder and it is therefore also necessarily present in the received

data stream. The same is true for the source memory. Whereas tandem decoding will ignore both the residual redundancy and the source memory, we can consider designing a joint source-channel decoder that would take either one or both sources of natural redundancy into account. In fact Shannon mentioned this possibility already in his 1948 paper [2] as part of the discussion on the implications of the source-channel theorem:

> "However, any redundancy in the source will usually help if it is utilized at the receiving point. In particular, if the source already has redundancy and no attempt is made to eliminate it[...], this redundancy will help combat noise."

Another advantage of such a joint decoding scheme lies in the fact that conventional source decoders, designed to perform the inverse operation of the source coder, cannot handle errors. In all practical systems, errors are still present at the channel decoder output/source decoder input and the source decoder's performance significantly decreases. This decrease in performance is further exacerbated in the case of VLC codes. This stems mainly from the fact that VLC codes do not have set symbol boundaries since the data symbols are encoded with variable bit lengths. One symbol error in the beginning of the data stream and the decoder may falsely estimate all the remaining symbol boundaries resulting in multiple decoding errors. It is in fact possible to partially resolve this problem in a joint decoding scheme. Recently, Miller [11] has shown that a joint decoder utilizing the source residual redundancy decreases this de-synchronization effect. Specifically, his work considers the case of Huffman encoding of a Markov source sent directly through a Binary Symmetric Channel. Miller's approach was later extended by Bauer [12], as he incorporated channel codes (specifically FEC codes) into the problem and again showed significant improvements with respect to the de-synchronization issue. We should note here however that Bauer used reversible variable length codes (RVLC's) which were introduced in [13] and have the advantage that the symbol boundaries may be recovered by decoding in both forward and backward directions. As such they assure far better synchronization, since the data stream will most likely be synchronized in its beginning and end.

1.3.2 Optimal joint decoding solution

The above arguments outline incontestable reasons for considering the problem of joint decoding as applied to systems that employ separate encodings: gains are necessarily possible. It was not until 1998 that the authors in [15] developed a generic solution to the joint decoding problem as defined in our setting. The fundamental premise of Murad et al.'s optimal joint decoding is that three elements of the coding chain in figure 1.3 have an equivalent graphical representation. Specifically, with no loss in generality, we can consider that each element is represented by a Finite State Machine (FSM). One can then build the product FSM of all three models that would hence

characterize the entire coding operation. From this point, all known decoding algorithms apply. This methodology is best explained with a simple example. Consider a finite-alphabet discrete source with memory 1. The source alphabet or equivalently the state-space of its corresponding FSM is given by $S_1 = \{A, B, C\}$. The FSM is depicted in figure 1.4 where we refrained, for simplicity, from specifying the transition probabilities that should quantify each arrow. If we

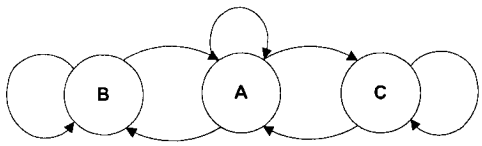


Fig. 1.4 FSM model of the assumed source.

assume further that $P(A) = 0.5$, $P(B) = 0.3$ and $P(C) = 0.2$, a block length 1 binary Huffman encoder may result in the assignment $A \rightarrow 0$, $B \rightarrow 10$ and $C \rightarrow 11$. Below, we show the placement of the source symbols on a binary tree, from which one can immediately derive a corresponding FSM representation of the Huffman encoding: this can be done by considering each one of the black vertices as a state. The state-space of the Huffman encoder FSM is $S_2 = \{X, Y\}$. Finally,

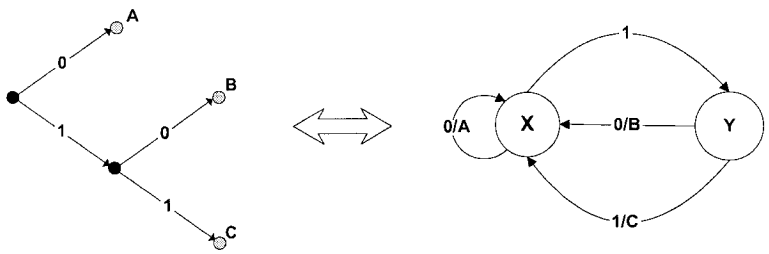


Fig. 1.5 FSM model of the Huffman encoding.

let us assume we are using a rate $\frac{1}{2}$ systematic convolutional encoder with generator polynomial $g(D) = (1,\ 1 + D)$, the method of obtaining a corresponding FSM is well-established [16]. The

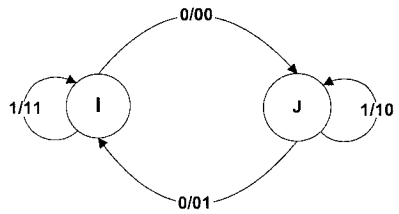


Fig. 1.6 FSM model of the channel coder.

state space of the channel encoder FSM is $S_3 = \{I, J\}$. With the FSM of every element in the

coding chain available, we may construct the *product* FSM of all three models. This is done by considering all possibilities of triplets of states and connecting them according to the rules dictated by each constituent model. Figure 1.7 below shows the product model of our example. Note that some states were redundant and hence they were removed. We also did not quantify the links for simplicity: the latter should be quantified by either deterministic transitions (from the Huffman coder and the channel coder) or probabilistic transitions (from the source model) along with appropriate outputs. With such a model available, various algorithms may be applied

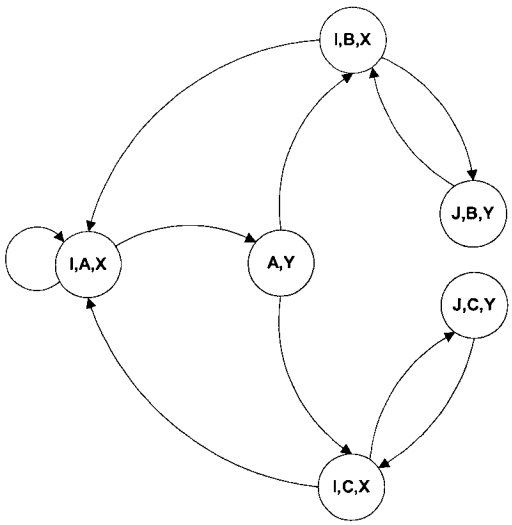


Fig. 1.7 Product FSM.

to yield the *optimal* joint decoding solution with respect to the algorithms' criteria. In particular, hard Viterbi, soft Viterbi, BCJR (or Kalman smoothing) and Kalman filtering are all readily applicable. Such a joint decoding scheme therefore uses three types of information: the source memory, the source coder residual redundancy and finally, the redundancy introduced by the channel coder.

1.3.3 Need for suboptimal joint decoding schemes

Unfortunately, the optimal solution remains intractable for most practical systems. In general, the state-space S_P of the product model satisfies $S_P \subseteq \{S_1 \times S_2 \times S_3\}$ and hence we have that $|S_P| \leq |S_1||S_2||S_3|$ where $\times$ denotes the Cartesian product. This state-space *explosion* is unaffordable in practical situations: a source alphabet, for say image transmission, will satisfy $|S_1| = 2^8$, the state-space of a VLC code will then conservatively reach $|S_2| = 2^8$, while the channel coder, if it uses 5 bits of memory will have $|S_3| = 2^5$, leading to a product model with $|S_p| = 2^{20}$ or more.

The need for less complex and therefore suboptimal decoding schemes, is flagrant.

In this context, the works of Bauer and Hagenauer in [17]–[18] provided a sub-optimal joint decoding under the assumption that the source is memoryless. Indeed, he considered the case of a general VLC code followed by a channel code — the two components separated via an interleaver. The proposed algorithm uses the principle of turbo-decoding and alternates the use of a VLC soft decoder with a soft channel decoder. His approach is particularly interesting since it has the advantage of isolating two soft-decoders and therefore, it has limited complexity. Since turbo-decoding is essentially sub-optimal decoding of a complex code [19], Bauer's inspiration is especially meaningful.

Finally, Guyader [20] et al. extended this approach to encompass all three elements of the coding chain and therefore include the source's memory. Their proposed sub-optimal algorithm, which this text is largely inspired from, relies as well on the principles of turbo-decoding and was developed in the context of Bayesian networks. It is important to also mention the related works of Villasenor et al. and Zhu et al. in [21]–[23] who also considered the problem of joint source-channel iterative decoding. However, whereas the works of Villasenor deals with systems that require the use of small packets, Zhu deals with systems with multiple channels or descriptions and as such, these works are not directly relevant to the topic at hand. Chapter 3 is reserved for the discussion of Guyader's algorithm. For now, a review of Bayesian networks is in order.

Chapter 2

Bayesian Networks and Belief Propagation

The original inspiration for Bayesian networks stems from an attempt to mimic human inferential reasoning within the natural frameworks of probability theory and graph theory. The original idea, as introduced by Judea Pearl [24], was that human knowledge, generally uncertain and incomplete, is stored not in joint distributions but rather in conditional distributions. This was thought to account for the relative ease with which we deal with statements such as *the probability of rain given a cloudy day* as well as the speed and reliability of human decisions involving similar statement. Pearl's Belief Propagation algorithm, developed in this context, in fact represents an efficient solution to the generalized inference or estimation problem; a solution that exploits the conditional dependence relations of the random variables involved. Although Bayesian networks were mainly to be limited to the field of Artificial Intelligence, researchers are finding various new applications for the idea such as data mining, and more importantly the problem of decoding that we are concerned with. This chapter begins with a rigorous definition and method of construction for Bayesian networks, followed by a discussion on some *emergent* properties of such networks. Next, we present in detail Pearl's Belief Propagation algorithm as applicable to polytrees and trees. We consider how various algorithms that solve the inference problem may be seen as particular instances of Belief Propagation. Finally, convergence issues with respect to Belief Propagation are presented and discussed.

2.1 Defining a Bayesian Network

Bayesian networks are directed acyclic graphs (DAG) in which the nodes represent random variables and the arcs, quantified by conditional probability measures, represent direct statistical

dependencies between the linked random variables. Strictly speaking, it should be noted that Bayesian networks are not graphs but rather *hyper-graphs* since their topology is augmented with a set of conditional probability measures. Networks of this sort can be used to equivalently represent the generic knowledge, as determined by a joint probability measure, of a given statistical problem. They may also be turned into a computational architecture to manipulate the addition of new knowledge. Specifically, if the network is not merely used to *store* knowledge, one can consider using the network's topology together with its corresponding conditional probability measures to define and direct computations necessary for incorporating new information. In this sense, Bayesian networks provide a graphical framework for the analysis of statistical problems. In the following we assume, unless otherwise specified, that the random variables are discrete.

The question of how one represents the generic knowledge of a statistical problem via the Bayesian networks framework arises. In particular, given a joint distribution, specified[1] by $P(x_1, x_2, \ldots, x_n)$, on the random variables $X_1, X_2, \ldots, X_n$, how does one determine the corresponding Bayesian network representation of this problem domain? It should be clear from the previous discussion that the nodes of the graphs are already available: the corresponding Bayesian network consists of a total of n nodes, one for each random variable X_i with i ranging from 1 to n. The arcs linking the random variables as well as the conditional probability measures are the only elements lacking for a full specification of the Bayesian network. Choosing an arbitrary ordering d on the random variables as $X_1, X_2, \ldots, X_n$, a recursive application of Bayes' law will yield the following relation:

$$P(x_1, x_2, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1) \cdots P(x_3|x_2, x_1)P(x_2|x_1)P(x_1) \tag{2.2}$$

In this expression, each factor contains only one variable on the left hand side of the conditioning bar and all conditional dependencies, assuming the ordering d, are represented. Therefore it may be used as a prescription for consistently determining the linking arcs together with the conditional probability measures. Specifically, for each factor, we may simply draw an arc emanating from each random variable on the right hand side of the conditioning bar and terminating at the random variable on the left hand side of the conditioning bar; this set of arcs is then quantified by the factor itself. If no arcs terminate at a given node, the latter is assigned an a-prior marginal distribution. For example, with the ordering specified above, there would be one arc from X_1 to X_2 quantified by the probabilities $P(x_2|x_1)$; one arc, from X_2 to X_3, and another, from X_1 to X_3, who together are quantified by the probabilities $P(x_3|x_2, x_1)$ and so on. Since no arc terminates

[1] note that the joint probability mass function is given by

$$p(x_1, x_2, \ldots, x_n) \triangleq P(X_1 = x_1, X_2 = x_2 \ldots, X_n = x_n) = P(x_1, x_2, \ldots, x_n) \tag{2.1}$$

at X_1 it is assigned the a-prior probabilities $P(x_1)$.

In general, equation 2.2 can be further simplified. It is indeed possible that given knowledge of X_2, X_3 is statistically independent of X_1. Expressed mathematically, $P(x_3|x_2, x_1) = P(x_3|x_2)$. In this case, in fact, only one arc from X_1 to X_3, quantified by $P(x_3|x_2)$, is necessary. This may be done for every factor. For example consider a joint distribution factoring according to,

$$\begin{aligned} P(x_1, x_2, \ldots, x_7) &= P(x_7|x_1, x_3, x_4)P(x_6|x_1, x_2, x_4) \\ & \quad P(x_5|x_1, x_2, x_3)P(x_4)P(x_3)P(x_2)P(x_1) \end{aligned} \tag{2.3}$$

The corresponding Bayesian network is shown below in figure 2.1.

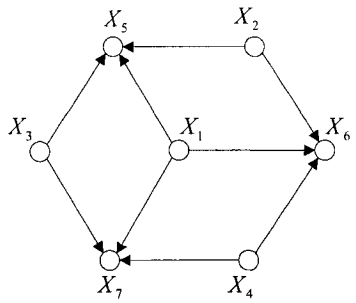


Fig. 2.1 Typical Bayesian network.

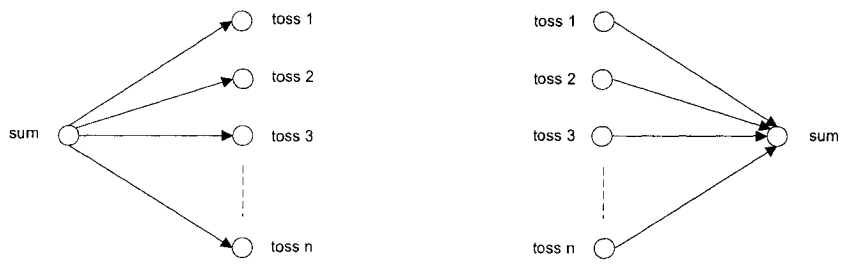
This, more generally, leads to a simple method for the construction of a Bayesian Network for *any* joint distribution. We start by imposing an arbitrary ordering d on the set of random variables[2], $X_1, X_2, \ldots, X_n$. We then choose X_1 as a root of the graph and assign it the marginal probabilities $P(x_1)$ as dictated by $P(x_1, x_2, \ldots, x_n)$. Next, we form node X_2; if X_2 is dependent on X_1, a directed link from X_1 to X_2 is established and quantified by $P(x_2|x_1)$. Otherwise, we leave X_1 and X_2 unconnected and assign the prior probabilities $P(x_2)$ to node X_2. At the ith stage, we form node X_i and establish a group of directed links to X_i from the smallest subset of nodes $S_i \subseteq \{X_1, X_2, \ldots, X_{i-1}\}$ satisfying the condition

$$P(x_i|S_i) = P(x_i|x_{i-1}, \ldots, x_1) \tag{2.4}$$

The links are then quantified by $P(x_i|S_i)$. Each element of S_i is called a *parent* of X_i while X_i is referred to as a *child* of each element of S_i and we may clearly write, $P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|S_i)$. It can be shown that the set of subsets satisfying condition 2.4 is closed under

[2]we assume for simplicity of notation and with no loss in generality that the chosen ordering is as indicated by the indices.

intersection [24], therefore the minimal subset S_i is *unique*. Thus, the joint distribution, specified by $P(x_1, x_2, \ldots, x_n)$, together with the ordering d uniquely identify a set of parent nodes for each variable X_i, and that constitutes a full specification of a directed acyclic[3] graph representing $P(x_1, x_2, \ldots, x_n)$. It is clear that different orderings will yield different factorizations of the joint distribution which in turn lead to significantly different Bayesian networks: a Bayesian network representing n independent coin tosses together with the modulo-2 sum of these tosses is turned from a tree to a full graph if we change the position of the sum variable from first to last as is shown in figure 2.2. However all the resulting Bayesian networks are equivalent in the sense that they encode the same joint distribution.



(a) sum variable is first in ordering. (b) sum variable is last in ordering.

Fig. 2.2 Bayesian network corresponding to n coin tosses and their modulo-2 sum.

2.2 Emergent Properties

From a mathematician's perspective, a Bayesian network representation of a statistical problem is utterly trivial. Indeed, the joint distribution of a set of random variables already contains all possible information of interest: any probabilistic question is readily available through an appropriate arithmetic manipulation of the joint distribution. However, as is the case with any framework, Bayesian networks posses emergent properties that may consolidate and simplify the understanding and analysis of statistical problems. In this section, we consider two such properties: the reduction in storage space for the representation of the joint distribution and necessarily computational savings when incorporating new information, as well as the representation of conditional independence and dependence relations.

[3] a simple proof by contradiction shows that the method of construction, for any ordering and any distribution, implies the absence of directed cycles.

2.2.1 Computational savings

Let us reconsider the example given above with the joint distribution satisfying the factorization of equation 2.3 and let us further assume that all random variables are binary. If we were to store the joint distribution directly, we would require $2^7 = 128$ entries. If we consider the Bayesian network representation that is based upon the factoring of the joint distribution into conditional distributions, we note that for a given node of k parents, a function of $k+1$ arguments is necessary for the specification of the conditional probability measures that quantifies the k links. Hence, we would require $3 \times 2^4 + 4 \times 2 = 56$ entries to equivalently store the same information, a significant decrease in storage[4]. More importantly, Bayesian network representations allow for computational savings when say computing posterior marginals given the instantiation of a set of random variables. Supposing for example that we wish to compute the quantity $P(x_7|x_1)$. Working from the joint distribution alone, one would apply Bayes' law and write $P(x_7|x_1) = P(x_7, x_1)/P(x_1)$, where both the numerator and denominator are obtained via the *law of total probability* by summing the joint distribution over all remaining variables. Such an approach therefore requires $2^5 + 2^6 = 96$ summation operations and 1 division operations for each pair (x_7, x_1). Using the Bayesian network representation, we note that X_7 is in fact only statistically dependent on X_1, X_3 and X_4. We may obtain $P(x_7|x_1)$ by summing the product $P(x_7|x_1, x_3, x_4).P(x_3).P(x_4)$, all of which are readily available, over X_3 and X_4. Hence only 8 multiplications and 4 summation operations are necessary for each (x_7, x_1) pair.

In general, storing a joint distribution requires a space growing exponentially with the number of random variables and the answer to queries regarding marginals, be they prior or posterior, is as well exponentially long with the number of variables. The point here is that, by exploiting statistical dependence and independence relations and storing conditional distributions, Bayesian networks allow for considerable savings with respect to both these issues.

2.2.2 Conditional independence relations

A significant emergent property of the Bayesian network framework is that the network's topology may be used to establish various types of conditional independence relations. Consider a triplet of random variables, X_1, X_2, X_3, where X_1 is connected to X_3 via X_2. The two links, connecting the pairs (X_1, X_2) and (X_2, X_3) can join at the midpoint X_2 in three possible ways:

1. Tail-to-Tail: $X_1 \leftarrow X_2 \rightarrow X_3$

[4]to be completely exact, storage space for the topology of the graph is also required. However with more random variables and particularly ones taking values on alphabets with greater cardinality, this storage space becomes quickly negligible.

2. Head-to-Tail: $X_1 \rightarrow X_2 \rightarrow X_3$ or $X_1 \leftarrow X_2 \leftarrow X_3$

3. Head-to-Head: $X_1 \rightarrow X_2 \leftarrow X_3$

Assuming that X_1, X_2, X_3 are the only variables involved, it should be clear from the aforementioned method of construction that in the first two cases, X_1 and X_3 are conditionally independent given X_2. Indeed in these two cases, X_2 cannot be the last variable in the imposed ordering: this position must have been filled by either X_1 or X_3 and since there is no link between the two, the previous statement immediately follows. In the last case, X_1 and X_3 are marginally independent: X_2 is necessarily the last variable in the ordering and since no link connects X_1 and X_3, we have $P(x_3|x_1) = P(x_3)$. However, X_1 and X_3 may become dependent given knowledge of X_2. Moreover, if X_2 has descendants $X_4, X_5, \ldots$, then X_1 and X_3 may also become dependent if one of those variables is known (instantiated). These considerations motivate definitions for a qualified notion of graph-separability sensitive to the directionality of the links and to all variables that are known as mentioned in [24].

Definition 2.1. Two links meeting Tail-to-Tail or Head-to-Tail at node X are *blocked* by a subset of variables S_e if $X \in S_e$. Two links meeting Head-to-Head at node X are *blocked* by S_e if neither X nor any of its descendants is in S_e.

Definition 2.2. A path P is *separated* by a subset S_e of variables if at least one pair of successive links along P is *blocked* by S_e.

Definition 2.3. S_e is said to *separate* X_i from X_j if all paths between X_i and X_j are *separated* by S_e.

where a path is defined as a sequence of nodes $\{X_1, X_2, ..., X_n\}$ such that the pairs $\{X_{i-1}, X_i\}$ are linked either as $X_i \rightarrow X_{i+1}$ *or* $X_i \leftarrow X_{i+1}$. This definition of separation provides a graphical criterion for testing conditional independence. It is in fact possible to prove [25] that if S_e separates X_i from X_j then X_i is conditionally independent of X_j given S_e. That is,

$$P(x_i|x_j, S_e) = P(x_i|S_e) \tag{2.5}$$

The implication of this statement being that one can visually determine a set of variables that would cause two other given variables to be conditionally independent. Moreover, this graph-separation criteria permits the identification by inspection of a *screening neighborhood* for any given node, namely, a set S_c of variables that renders a given variable independent of every variable not in S_c. Indeed the union of the the following three types of neighbors is sufficient for forming a screening neighborhood: direct parents, direct children and all direct parents of the

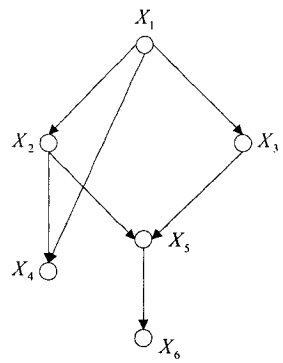


Fig. 2.3 Bayesian network example representing the distribution specified by $P(x_1, \ldots, x_6) = P(x_6|x_5)P(x_5|x_2, x_3)P(x_4|x_1, x_2)P(x_3|x_1)P(x_2|x_1)P(x_1)$

latter as dictated by the above definitions. Considering for example the Bayesian network shown in figure 2.3, we note that X_2 and X_3 are separated by $S_e = \{X_1, X_4\}$ since the two paths between X_2 and X_3 are blocked and hence $P(x_2|x_3, x_1, x_4) = P(x_2|x_1, x_4)$. Such a relation is read with ease off the graph but would imply a significant amount of arithmetic tedium if it were to be proven from the joint distribution. Note that a screening neighborhood of X_3 is $S_c = \{X_1, X_5, X_2\}$.

It is important to note that although graph-separability implies conditional independence, the converse is by no means true. Since the structure of a Bayesian network is heavily dependent on the node ordering, not all conditional independence relations are made transparent by the graph's topology: networks corresponding to particular orderings may very well express graphical separability conditions that are not graphically valid for networks with different orderings. Therefore a particular Bayesian network does not provide a *complete* characterization of all conditional independence relations via the graph separability definition. However since conditional independence is a property of the underlying distribution and therefore order-invariant, those relations that do become transparent under a particular ordering remain valid under all other ordering eventhough a graph-separation is not induced. For a rigorous discussion of the above, the reader is referred to [25]. On a final and brief note, the graph-separation criterion is extremely useful when attempting to model complex statistical problems in the Bayesian network framework. Consider for example building a Bayesian network corresponding to a medical *expert system* that is to model the interactions between all known symptoms and all known diseases. A joint distribution for this problem is hardly available. What is available however is expert opinion on which symptoms may be expressed given a disease. Therefore, we may build the network, making sure graph-separability holds where it must and all that remains, to consistently solve this problem, is defining the appropriate conditional probabilities [26].

2.3 Pearl's Belief Propagation

As previously mentioned, once a Bayesian network is constructed, it can not only be used to represent the generic knowledge of a given domain, but can also be consulted to calculate the impact of specific input data on some nodes (random variables). This process, essentially involves the instantiation of a subset of nodes and the subsequent calculation of posterior marginals for those remaining nodes of interest. In general, this process may be guided by an external interpreter that has knowledge of the entire network and would therefore select and direct calculations. However, the algorithm presented in this section, termed Belief Propagation and originally introduced by Judea Pearl [24], assumes no such interpreter. In fact, the network's topology is seen as providing a computational architecture allowing the incorporation of new information as represented by the instantiation of a set of nodes. As such, the links of the network are treated as *pathways* for directing the flow of data in the updating of probabilities and the nodes of the network are treated as *activation centers* that propel the entire process. Accordingly, it is assumed that each node in the network is designated a separate processor responsible for two tasks: maintaining the current probabilistic information pertaining to its host variable and managing the communication links to the set of neighboring nodes. The communication links are assumed *open* at all times so that each processor may at any time verify whether its own information corresponds with that provided by its neighbors: if the information agrees, no activity takes place, otherwise the node activates its update mechanism. In the next subsections, we show in the details the working of this algorithm for various classes of graphs.

2.3.1 Discrete Polytrees

Here we assume that all random variables are discrete and we further assume that that the resulting Bayesian network is *singly connected*[5]. Finally, we suppose a set of leaf nodes[6] have been instantiated and denote the total evidence obtained by $\mathbf{e}$. We wish to compute the posterior marginal probabilities of all remaining nodes. We consider a typical fragment of a singly connected network as shown in figure 2.4. We denote $\mathbf{e}_X^-$, the evidence connected to the random variable X, with instance x, through the set of its children $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_m\}$, and $\mathbf{e}_X^+$ the evidence connected to X through its set of parents $\mathbf{U} = \{U_1, U_2, \ldots, U_n\}$.

We use $BEL(\cdot)$ as shorthand notation[7] for the current posterior marginal probability $P(\cdot|\mathbf{e})$

[5] namely, no more than one path exists between any two nodes.

[6] this assumption is simply to avoid cumbersome notation and comes with no loss in generality as will be seen.

[7] we refer to this quantity as the *belief* of a random variable.

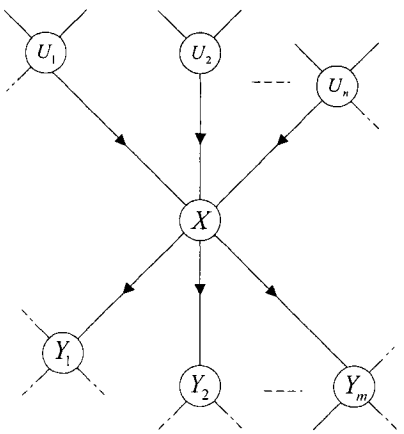


Fig. 2.4 Parents and children of a typical node X in a singly connected network.

so that a simple application of Bayes' law yields

$$\begin{aligned} BEL(x) &= \alpha P(\mathbf{e}_X^-|x)P(x|\mathbf{e}_X^+) \\ &= \alpha\lambda(x)\pi(x) \end{aligned} \tag{2.6}$$

where $\alpha = [P(\mathbf{e}_X^-|\mathbf{e}_X^+)]^{-1}$ is a normalizing real constant. Here $\lambda(x)$ represents the retrospective information[8] that X receives from its descendants and $\pi(x)$ represents causal information by all non-descendants of X, mediated by X's parents. Now, $\mathbf{e}_X^-$ and $\mathbf{e}_X^+$ can be further decomposed into

$$\mathbf{e}_X^- = \{e_{XY_1}^-, \ldots, e_{XY_m}^-\} \qquad \text{and} \qquad \mathbf{e}_X^+ = \{e_{U_1X}^+, \ldots, e_{U_nX}^+\} \tag{2.7}$$

where $e_{XY_j}^-$ stands for the evidence contained in the subnetwork on the *head* side of $X \rightarrow Y_j$, and $e_{U_iX}^+$ stands for the evidence in the subnetwork contained on the *tail* side of the link $U_i \rightarrow X$. Now, to see how information from several descendants may be combined, we note,

$$\begin{aligned} \lambda(x) &\triangleq P(\mathbf{e}_X^-|x) \\ &= P(e_{XY_1}^-, \ldots, e_{XY_m}^-|x) \\ &= P(e_{XY_1}^-|x) \cdot P(e_{XY_2}^-|x) \cdots P(e_{XY_m}^-|x) \\ &= \prod_{j=1}^{m} \lambda_{Y_j}(x) \end{aligned} \tag{2.8}$$

[8] note that $\lambda(x) = P(\mathbf{e}_X^-|x)$ is used to denote the probability of the data or evidence $\mathbf{e}_X^-$ given X=x, and should be understood to be a function of x.

where $\lambda_{Y_j}(x) = P(e^-_{XY_j}|x)$ and where the second equality follows from the fact that graph-separation implies that Y_i is conditionally independent from Y_j given X for $i \neq j$ and hence any evidence contained in these nodes' corresponding subnetworks is as well conditionally independent given X. Therefore $\lambda(x)$ may be computed from information present within its descendants. To see how X may compute its $\pi(x)$ vector from information contained within its parents, we note,

$$\begin{aligned}
\pi(x) &\triangleq P(x|\mathbf{e}^+_X) \\
&= P(x|e^+_{U_1X}, \ldots, e^+_{U_nX}) \\
&= \sum_{u_1,u_2,\ldots,u_n} P(x|u_1,u_2,\ldots,u_n) \cdot P(u_1,u_2,\ldots,u_n|e^+_{U_1X},\ldots,e^+_{U_nX}) \\
&= \sum_{u_1,u_2,\ldots,u_n} P(x|u_1,u_2,\ldots,u_n) \cdot P(u_1|e^+_{U_1X}) \cdot P(u_2|e^+_{U_2X}) \cdots P(u_n|e^+_{U_nX}) \qquad (2.9)
\end{aligned}$$

where the second equality follows from the law of total probability and the third equality follows from the fact that each pair $\{U_i, e^+_{U_iX}\}$ is independent of $\{U_j, e^+_{U_jX}\}$ for $i \neq j$ (see previous section on graph separation). Letting, $\pi_X(u_i) = P(u_i|e^+_{U_iX})$, we can write,

$$\pi(x) = \sum_{\mathbf{u}} P(x|\mathbf{u}) \prod_{i=1}^{n} \pi_X(u_i) \qquad (2.10)$$

Substituting equation 2.8 and equation 2.10 into equation 2.6, we have:

$$BEL(x) = \alpha \left[\prod_{j=1}^{m} \lambda_{Y_j}(x)\right] \left[\sum_{\mathbf{u}} P(x|\mathbf{u}) \prod_{i=1}^{n} \pi_X(u_i)\right] \qquad (2.11)$$

Therefore, node X may compute its belief (posterior probability) if it receives messages $\lambda_{Y_j}(x)$ from its children and $\pi_X(u_i)$ from its parents. We must now define how a typical node, say X, will compute its outgoing messages $\lambda_X(u_i)$ and $\pi_{Y_j}(x)$ from the incoming messages $\lambda_{Y_j}(x)$ and $\pi_X(u_i)$ with $i = 1, \ldots, n$ and $j = 1, \ldots, m$. It is convenient to temporarily treat all parents of X except for U_i as a single compound variable $V = \{U_1, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n\}$ connected to X via a single link $V \rightarrow X$, as shown in figure 2.5.

Consider the message $\lambda_X(u_i)$ which node X must send its parent U_i so that the latter may in its turn update its belief. By definition, we have, $\lambda_X(u_i) = P(\mathbf{e}^-_{U_iX}|u_i)$. Now the evidence $\mathbf{e}^-_{U_iX}$ can be decomposed into two components: $\mathbf{e}^-_{U_iX} = \{\mathbf{e}^+_{VX}, \mathbf{e}^-_X\}$ where $\mathbf{e}^+_{VX} = \bigcup_{k \neq i} \mathbf{e}^+_{U_kX}$, therefore

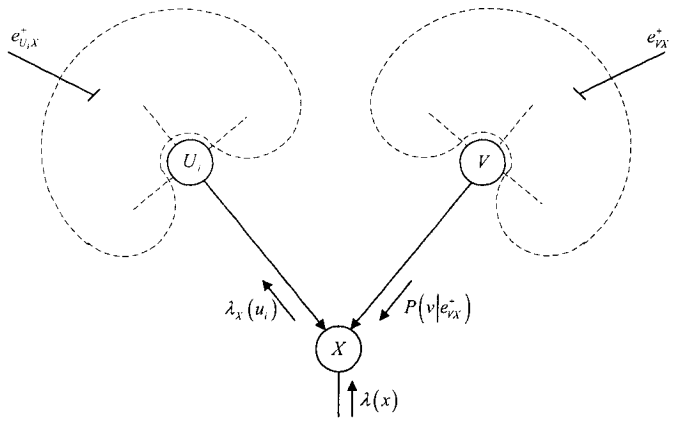


Fig. 2.5 Messages and evidence sets used in derivation of $\lambda_X(u_i)$.

we now have,

$$\begin{aligned}
\lambda_X(u_i) &= P(\mathbf{e}^+_{VX}, \mathbf{e}^-_X|u_i) \\
&\overset{(1)}{=} \sum_x \sum_v P(\mathbf{e}^+_{VX}, \mathbf{e}^-_X|u_i, v, x)P(v, x|u_i) \\
&\overset{(2)}{=} \sum_x \sum_v P(\mathbf{e}^-_X|x)P(\mathbf{e}^+_{VX}|v)P(v, x|u_i) \\
&\overset{(3)}{=} \beta \sum_x \sum_v P(\mathbf{e}^-_X|x)\frac{P(v|\mathbf{e}^+_{VX})}{P(v)}P(x|v, u_i)P(v|u_i) \\
&\overset{(4)}{=} \beta \sum_x \sum_v P(\mathbf{e}^-_X|x)P(v|\mathbf{e}^+_{VX})P(x|v, u_i)
\end{aligned} \tag{2.12}$$

where β is a real normalizing constant; (1) is given by the law of total probability; (2) follows since X separates $\mathbf{e}^-_X$ from $\mathbf{e}^+_{VX}$ and since V separates $\mathbf{e}^+_{VX}$ from U_i; (3) follows from Bayes' law; and (4) follows since U_i and V are marginally independent: $P(U_i|V) = P(U_i)$. Now, ungrouping the parents V and using $\pi_X(u_k)$ as previously defined, we have,

$$P(v|\mathbf{e}^+_{VX}) = \prod_{k\neq i} P(u_k|\mathbf{e}^+_{VX}) = \prod_{k\neq i} P(u_k|\mathbf{e}^+_{U_kX}) = \prod_{k\neq i} \pi_X(u_k) \tag{2.13}$$

and noting that $\lambda(x) = P(\mathbf{e}^-_X|x)$ as defined, and that $\{v, u_i\} = \mathbf{u}$, $\lambda_X(u_i)$ becomes

$$\lambda_X(u_i) = \sum_x \lambda(x) \sum_{u_k:k\neq i} P(x|\mathbf{u}) \prod_{k\neq i} \pi_X(u_k) \tag{2.14}$$

Consider now the message $\pi_{Y_j}(x)$ which node X must send to its child Y_j. By definition, we have, $\pi_{Y_j}(x) = P(x|e^+_{XY_j})$. Now $e^+_{XY_j}$ represents the evidence in the entire network with the exception of the evidence present in the subnetwork on the head side of the link $X \rightarrow Y_j$: $e^+_{XY_j} = \mathbf{e} - e^-_{XY_j}$. Therefore, $\pi_{Y_j}(x) = P(x|e^+_{XY_j})$ is in fact equal to $BEL(x)$ when the evidence $e^-_{XY_j}$ is omitted. Following the same lines as the previous derivation of $BEL(x)$, we get,

$$\pi_{Y_j}(x) = \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x) = BEL(x)\Big|_{\lambda_{Y_j}(x)=\mathbf{1}} \tag{2.15}$$

This equation interestingly indicates that an incoming message, $\lambda_{Y_k}(x)$, on a link will not affect the outgoing message, $\pi_{Y_j}(x)$, on the same link.

Summary of Belief Propagation for polytrees

The belief of node X can be computed if three parameters are available: messages $\pi_X(u_i) = P(u_i|e^+_{U_iX})$ from each parent U_i; messages $\lambda_{Y_j}(x) = P(e^-_{XY_j}|x)$ from each child Y_j; and finally the conditional probability matrix $P(x|u_1,\ldots,u_n) = P(x|\mathbf{u})$ relating X to its parent set. Local updating may be essentially implemented in three steps.

STEP 1—Belief updating

The belief of X is given below with α such that $\sum_x BEL(x) = 1$,

$$\begin{aligned} BEL(x) &= \alpha\lambda(x)\pi(x) \\ &= \alpha\left[\prod_{j=1}^{m}\lambda_{Y_j}(x)\right]\left[\sum_{\mathbf{u}}P(x|\mathbf{u})\prod_{i=1}^{n}\pi_X(u_i)\right] \end{aligned} \tag{2.16}$$

STEP 2—Bottom-up propagation

Message $\lambda_X(u_i)$ to be sent to parent U_i is given below with β such that $\sum_{u_i}\lambda_X(u_i) = 1$,

$$\lambda_X(u_i) = \beta\sum_x\lambda(x)\sum_{u_k:k\neq i}P(x|\mathbf{u})\prod_{k\neq i}\pi_X(u_k) \tag{2.17}$$

STEP 3—Top-down propagation

To compute message $\pi_{Y_j}(x)$ to be sent to child Y_j

$$\pi_{Y_j}(x) = \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x) \tag{2.18}$$

These three steps may be executed by a node in any desired order and constitute complete and consistent local operations that will eventually lead all nodes to their correct posterior probabilities: recalling that the original assumption that communication links are open at all times, nodes that detect changes in their belief values will trigger their update algorithm (steps 2 and 3) and the graph will eventually reach equilibrium with no further updates necessary. We have yet to establish boundary conditions that will allow the proper functioning of the algorithm. Without further ado, we provide the boundary conditions:

1. *Root nodes*: If X is a node with no parents, we set $\pi(x)$ to be equal to the prior distribution $P(x)$.
2. *Uninstantiated leaf nodes*: If X is a childless node that has not been instantiated, we set $\lambda(x) = (1, 1, \ldots, 1)$.
3. *Instantiated nodes*: If X is any node and evidence $X = x'$ is obtained, we set $\lambda(x) = \delta_{x,x'} = (0, \ldots, 0, 1, 0, \ldots, 0)$ with a 1 at the x'-th position.

Boundary condition 3 merits some explanations. We recall that in the derivation it was assumed that only leaf nodes were to be instantiated. However this assumption does not affect the generality of the algorithm because the fact that X is say an evidence node, with value x', can be represented by instantiating a child node Z, representing a noiseless observation and therefore delivering a message $\lambda_Z(x)$ to X,

$$\lambda_Z(x) = \delta_{x,x'} = \begin{cases} 1 & x = x' \\ 0 & x \neq x' \end{cases}$$

from which boundary condition 3 immediately follows.

Finally, it is convenient at this point to consider the number of computations necessary for each node activation as represented by the three above steps. We assume for simplicity that all parent nodes of X host variables taking values on the same set of cardinality $|\mathcal{U}|$. Equivalently, we assume all child nodes to take values on a set of cardinality $|\mathcal{Y}|$. Finally, with X taking on values on a set of cardinality $|\mathcal{X}|$, the total[9] number of operations necessary for a node activation were found to be,

$$\begin{cases} n^2|\mathcal{U}|^n|\mathcal{X}| + n|\mathcal{U}|\big(|\mathcal{X}| + 1\big) + |\mathcal{X}|(m+1) + m^2|\mathcal{X}| & \text{multiplications} \\ (n+1)|\mathcal{U}|^n|\mathcal{X}| + m|\mathcal{X}| - (m+n+1) & \text{additions} \end{cases} \tag{2.19}$$

[9]here, our derivation assumes equations 2.16, 2.17 and 2.18 are to be implemented as given and we include as well the operations necessary for normalization. Indeed, it was noted by the author that normalization is necessary for stability in a software implementing the Belief Propagation algorithm. See Appendix A.

2.3.2 Discrete Trees

In this section, we consider Belief Propagation for trees. In terms of our previous assumption, trees are singly connected networks where in addition each node is allowed, at most, one parent. Therefore equations 2.16, 2.17 and 2.18 remain entirely valid. Denoting by U, the single parent of X and now using superscripts to denote the *values* taken by a random variable, we form the matrix M, where

$$[M]_{ij} = P(x^{(j)}|u^{(j)}) \tag{2.20}$$

Belief Propagation as obtained by reducing equations 2.16, 2.17 and 2.18 to the case where one parent is allowed is succinctly depicted below in figure 2.6.

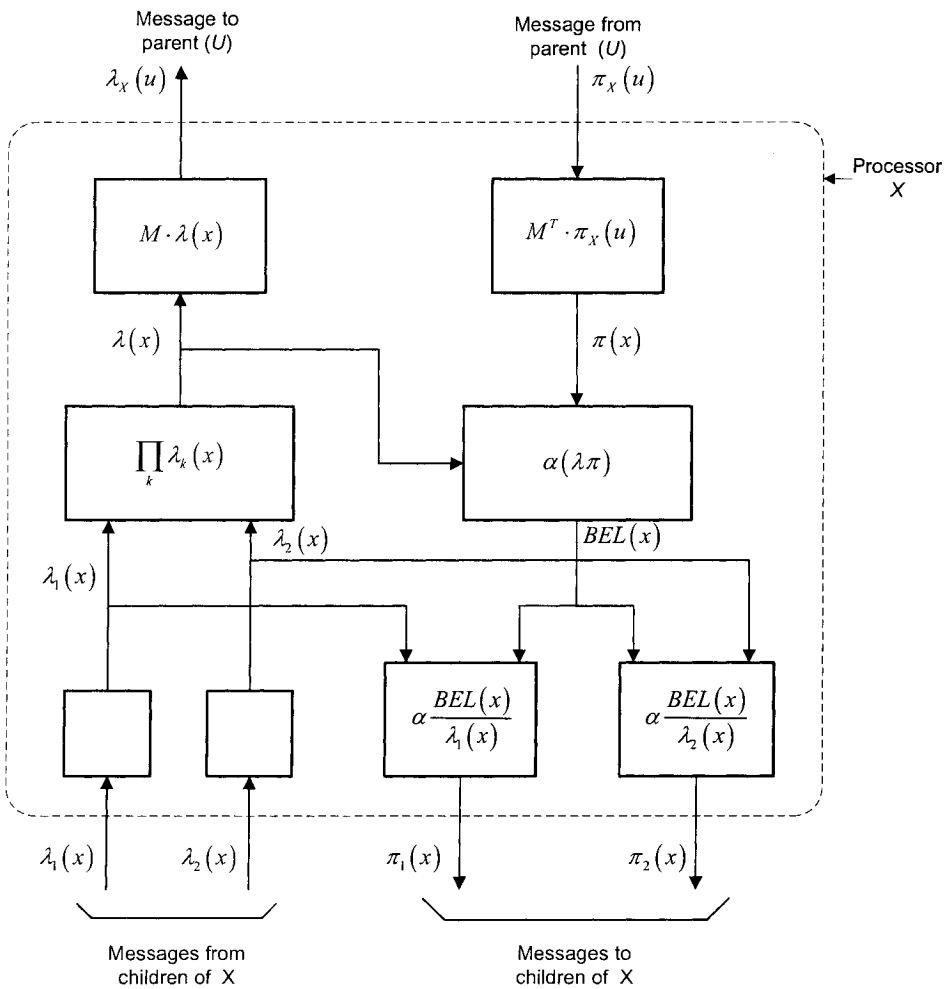


Fig. 2.6 Belief Propagation in trees.

2.3.3 Continuous observation on a discrete node

In this section we consider how a continuous observation on a discrete node may be incorporated into the Belief Propagation algorithm. Although this is not found in the literature, the derivation is straightforward. We assume that the discrete variable X has observation Z,

$$Z = aX + W \tag{2.21}$$

where $a \in \mathbb{R}$, and where W is a unidimensional gaussian random variable and therefore fully characterized by its mean m_W and variance σ_W^2: $W \sim (m_W, \sigma_W^2)$. The corresponding Bayesian network is shown in figure 2.7 below. In general X may be connected to a network however for simplicity we only show the nodes involved. Since Z is simply an *observation* node, we need not

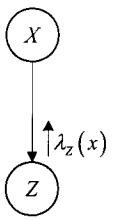


Fig. 2.7 Gaussian observation on discrete node.

be concerned with messages that it *receives* from X but rather the message $\lambda_Z(x)$ that it will *send* to its parent X. It should be clear from section 2.3.1 that if Z is not instantiated,

$$\lambda_Z(x) = (1, 1, \ldots, 1) \tag{2.22}$$

In the case that the event $Z = z_0$ occurs, we must use a modified definition of $\lambda_Z(x)$, namely that it represents the *likelihood* $L(\cdot)$ of the data e_{XZ}^-, given x:

$$\lambda_Z(x) = L(e_{XZ}^-|x) \tag{2.23}$$

where we recall that e_{XZ}^- stands for the evidence on the head side of the link $X \rightarrow Z$ and therefore is in fact the evidence contained in Z. Hence, we may write,

$$\lambda_Z(x) = L(Z = z_0|x) \tag{2.24}$$

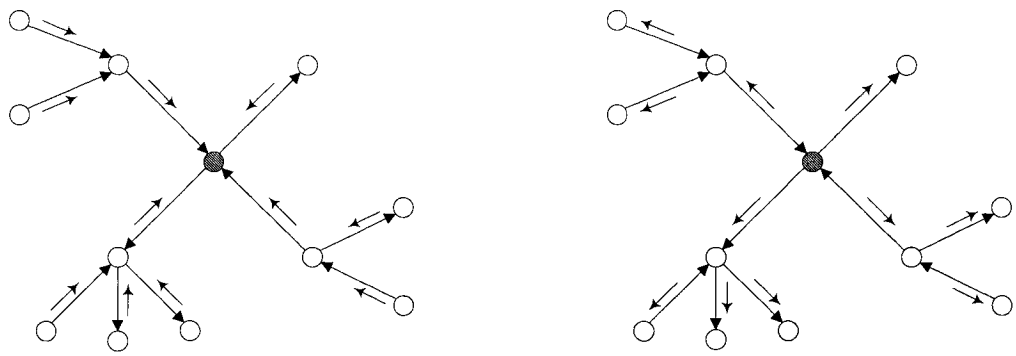
It is clear that given $X = x$, $Z \sim (ax + m_W, \sigma_W^2)$ and hence, with α such that $\sum_x \lambda_Z(x) = 1$, we may write,

$$\lambda_Z(x) = \frac{\alpha}{2\pi\sigma_W^2} e^{-\frac{1}{2\sigma_W^2}(z_0 - ax - m_W)^2} \tag{2.25}$$

2.4 More on Belief Propagation

2.4.1 Organized strategies

The Belief Propagation algorithm presented in the previous section will result in information being passed locally from one node to the other and is guaranteed to converge provided the assumption of a singly connected graph is not violated. The convergence time is proportional to the network diameter. It is also possible to externally direct node activations and so to speak organize the computations necessary for the network to reach equilibrium. In general this can be done with relative ease. Simply choose a node in the network and designate it the *center* node. Graph equilibrium can then be reached in two steps: bringing all the information to the center node where it is combined and subsequently redistributed to the rest of the network. Figure 2.8 shows this process. Belief Propagation in two steps has the disadvantage that knowledge of the entire



(a) Bringing information *in* to the center node.

(b) Bringing information out of the center node.

Fig. 2.8 Belief Propagation in two steps.

network's topology is required and hence breaking in spirit the assumption that computations are to be local. However it does provide the advantage at the end of the second step, all nodes are guaranteed to have appropriate posterior distributions. We note that belief updating in two steps in nothing more than a particular organization of computation of Pearl's Belief Propagation.

2.4.2 Belief Propagation and the inference problem

In the context of the general inference problem, namely the problem of estimating the values for a set of unobserved random variables given some data, Belief Propagation is in fact nothing more than an efficient, graphically based solution to the problem. Referring back to equation 2.19 and assuming all alphabet cardinalities of the random variables to be equal to q, we note that Pearl's algorithm solves the inference problem on singly connected networks with $O(q^{e+1})$ computations

where e is the maximum number of parents of any node. This stands in sharp contrast with the $O(q^m)$ computations, where m is the number of unknown random variables, which is required by the brute-force method of working from the joint distribution.

Interestingly, other algorithms which also solve the inference problem turn out to be particular instances of Belief Propagation. Consider for example the hidden Markov Chain problem, where an unobserved Markov Process X is to be estimated from its corresponding point-wise noisy observation process Y. The appropriate Bayesian network is shown in figure 2.9. Applying

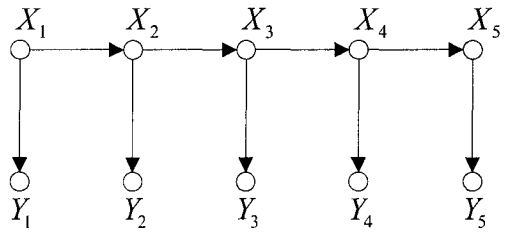


Fig. 2.9 Bayesian network for the hidden Markov chain problem.

Belief Propagation in two passes with X_5 as a center node will result in a linear-time exact solution which is functionally entirely identical to the BCJR algorithm [31]. As a final step, similarly to the BCJR algorithm, the posterior marginals computed by Belief Propagation may be used to provide maximum-a-posteriori (MAP) estimates for each X_i given the observations $Y = y$, that is $\hat{X}_i = \arg\max_x P(X_i = x|y)$. All of the above applies as well to Kalman smoothing. A particular organization, on the other hand of node activation will yield the Kalman filtering solution, namely if the nodes are activated in the following order: Y_1, X_1, Y_2, X_2,...,Y_5, X_5. Similarly, Kalman prediction of order τ turns out to be a particular organization of node activation: activate Y_i, X_i for $i = 1 \ldots t$ and finally activate X_{t+1} to $X_{t+\tau}$. Finally a small modification pertaining to the messages and update rules of Belief Propagation will result in a solution, equivalent to the Viterbi algorithm.

Belief Propagation seems therefore to be a generalization of the *forward-backward algorithm.* In the context of decoding, other algorithms have also been shown to be particular instances of Belief Propagation. In particular, McEliece, MacKay and Cheng [32] have recently shown the surprising yet intuitively meaningful relation between turbo (iterative) decoding and Belief Propagation. Specifically, it was shown that if Belief Propagation is applied to the Bayesian network corresponding to a parallel concatenation of two or more codes, the turbo decoding algorithm immediately results. McEliece also shows that the same connection holds for other previously known iterative algorithms.

2.4.3 Graphs with loops

In general, during the construction of a Bayesian network, it is possible that undirected cycles (loops) are formed. An undirected cycle is simply defined as a path in which the first node corresponds to the last. In such a situation the network is no longer singly connected and hence Belief Propagation is not guaranteed to yield correct posterior marginals. Indeed, the derivation of equations 2.16, 2.17 and 2.18 heavily depends on the assumption that evidence obtained from different parents is *independent* and the same applies for evidence obtained from different children.

There are essentially three methods that would allow us to cope with loops in such a situation and still compute correct posterior marginals. The first, *node aggregation*, collapses a set of particular nodes into one, so that a particular loop may be broken. In the example of figure 2.3, we may collapse nodes X_2 and X_3 into a single node representing (X_2, X_3) and the network becomes singly connected. This method works well on small loops but requires exponential storage space with the number of compounded variables. The second, *stochastic relaxation*, assumes that each processor examines the states of the nodes within its screening neighborhood, computes its belief, then randomly selects one of these values with the computed probability. The value chosen is then interrogated by the neighbors upon computing their beliefs, and so on. This scheme requires a very long time before reaching steady state. Finally, the third, *conditioning*, is based on rendering the network singly connected by instantiating a selected group of variables: as many networks as possible values of the selected group are created, Belief Propagation is carried out on each of those networks and the results are finally combined. This solution suffers from combinatorial explosion. For a rigorous discussion of the above the reader is referred to [27]

If on the other hand, we ignore the existence of loops and apply Pearl's Belief Propagation algorithm, messages may circulate indefinitely around the graph and applying the two step strategy will generally result in incorrect posterior distributions. However, in some situations, say for calculating the posterior marginal distribution of a bit, one does not necessarily need the exact distribution, as long as the final hard decision is correct. Therefore, in some cases, one can simply ignore the presence of loops and carry on with Belief Propagation. This misunderstood phenomenon is explored in detail in [28] and [29].

Chapter 3

Joint Source-Channel Decoding via Bayesian Networks

In the first chapter, we saw that a joint decoding strategy applied to systems that employ separate encodings will necessarily result in gains with respect to a tandem decoding strategy: particularly so in the case where a variable length source code is used. The optimal joint decoding solution was exposed and the flagrant need for sub-optimal joint decoders became apparent. In the last chapter, Bayesian networks were seen to provide a convenient graphical framework for the analysis of statistical problems. Belief Propagation, on the other hand, was shown to provide an efficient solution to the general inference problem and in the context of decoding, we saw that Belief Propagation may be used to yield MAP estimates of the quantities of interest. Here, we show how the joint decoding problem may be approached and analyzed within the framework of Bayesian networks. The discussion is based on the developments of Guyader et al. [20] who originally tackled the problem. However an attempt was made to reformulate and expande upon their ideas and derivation, for completeness and for the purposes of better understanding the algorithm proposed in the subsequent chapter. We begin this chapter with a brief section reiterating the problem of joint decoding as defined in our setting. Next, we show how the Bayesian network representation of the entire coding chain may be derived. A section expanding on the possibility of adapting the derived graph for the purposes of decoding under various restrictions follows. Finally, we show how iterative decoding may be applied on the resulting graph to yield a sub-optimal yet robust joint decoding algorithm applicable to both CLC's and VLC's.

3.1 The Joint Decoding Problem

We recall that in our predefined setting, of systems employing separate encodings, the joint decoding problem reduces to that of providing an estimate of the transmitted data based on the redundancy introduced by the channel coder and on either one or both the residual redundancy of the source coder and the source memory. Hence, figure 3.1 is the paradigm of our discussion. Our general assumption of a discrete finite-alphabet source is maintained. The source produces a

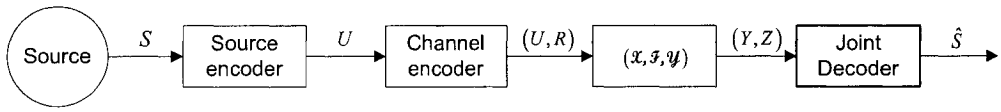


Fig. 3.1 The joint decoding problem.

symbol sequence S, which is in turn mapped via a binary source coder to a sequence of (information) bits U. The bits are sent to a channel coder, possibly systematic, producing the sequence (U, R) where R is the sequence the redundant bits. The sequence (U, R) is subsequently sent through a channel producing the observation sequence (Y, Z). The only assumptions that we impose on the channel are that it admits a binary input alphabet and that its set of conditional probability measures is causal: hence we are restricting ourselves to non-feedback, binary-input channels. The design of the last element, producing an estimate $\hat{S}$ of S, is the point of concern. We note that in the case that the channel coder is a non-systematic one, we may simply drop U and its corresponding observation Y from (U, R) and (Y, Z) respectively.

3.2 Deriving the Bayesian Network Representation of the Coding Chain

In this section, we will show the detailed derivation of the Bayesian network corresponding to the entire coding operations under the aforementioned assumptions.

3.2.1 Preliminaries

We will further assume that the source is given by a first order, stationary Markov process[1] generating symbols $S = S_1, S_2, \ldots, S_N$. We assume in addition that the source symbols are mapped via a block length one[2], binary source coder into a sequence of information bits $U = U_1, U_2, \ldots, U_K$. We denote by $\overline{U}_n$ the codeword corresponding to S_n. Note that we have not specified whether the source coder is of the CLC or the VLC variety. In the CLC case with codewords of length, l, we have that $K = Nl$, whereas in the VLC case, K is in fact a random variable given knowledge of N and vice-versa. Hence, in general, we note the presence of two time indices that are not

[1,2] these additional assumptions on the source and source coder may be relaxed as will be seen later

deterministically related: the symbol clock index, denoted by n, and the bit clock index, which we denote by k.

Deriving the Bayesian network corresponding to the Markov source and the source coder for the *symbol clock* is relatively straightforward. Simply consider the natural ordering imposed by the symbol clock time index, namely $S_1, \overline{U}_1, S_2, \overline{U}_2, \ldots, S_N, \overline{U}_N$. This ordering results in the following factorization of the joint distribution,

$$P(s_1, \ldots, s_n, \overline{u}_1, \ldots, \overline{u}_n) = P(s_1)P(\overline{u}_1|s_1) \prod_{n=2}^{N} P(\overline{u}_n|s_n)P(s_n|s_{n-1}) \tag{3.1}$$

The corresponding Bayesian network is shown in figure 3.2. Unfortunately, attempting inference

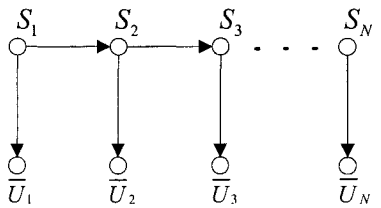


Fig. 3.2 Symbol clock model for the Markov source and source coder.

on such a graph leads to inescapable difficulties. First and foremost, knowledge of the transmitted symbol sequence length, N, is required. Second, in the VLC case, each $\overline{U}_n$ represents an *unkown* variable number of bits. Hence the very structure (topology) of the Bayesian network, if we were to expand $\overline{U}_n$ into its corresponding bit sequence, is random. For these reasons, it becomes much more convenient to derive the appropriate Bayesian network based on the *bit clock* time index. In that case, as will be seen, knowledge of the received bit sequence length, K, is required — a less restrictive assumption for the later context of decoding — and the topology of the Bayesian network, given K, is entirely deterministic. In the following, we derive the Bayesian network of the entire coding chain based on the bit clock time index.

3.2.2 The Markov source and source coder

To design the bit clock Bayesian network corresponding to the Markov source and the source coder, we must focus on U and analyze the structure of its distribution. This was essentially achieved in section 1.3.2, for the particular example the Markov source of three symbols. Indeed, the state-space representation of the source and that of the source coder may be combined in a single product state-space model. The result is an FSM with the information bits U_i depending on the transition from one state to another. Expanded in time, an order one Markov process is

obtained, once again with the information bits depending on the transition of states.

We now derive this process formally for a general order one Markov source and a general source coder. The state-space representation of the source is readily available: it consists of the set $\mathcal{S}$ of possible source symbols, namely the source alphabet, and the transitions from one state to another is given by the family of source transition probabilities $P(s_n|s_{n-1})$. A natural starting point is then to derive the state-space representation of the source coder. Recalling the definition of a binary source coder as an injective mapping from a symbol space to the power set of $\{0,1\}$, we let τ be the binary tree representing the source coder mapping, where a transition *upwards* corresponds to a codeword or information bit of 1 and a transition *downward* corresponds to a 0 bit. We begin first, for simplicity, by overspecifying the source coder's state-space and define it as the set $\mathcal{V}$ of all vertices of τ where a transition from one vertex (state) to the next produces the appropriate information bit. Now, we define the state-space $\mathcal{X}$ of the *product* Markov source and source coder model as $\mathcal{X} = \mathcal{S} \times \mathcal{V}$. The corresponding state variable is given by $X = (\Gamma, V)$ where Γ, with instance $\gamma^{(i)} \in \mathcal{S}$, is a variable representing the last completed symbols and V, with instance $v^{(j)} \in \mathcal{V}$, is variable representing the current vertex of τ describing the construction of the next symbol. The state transition probabilities are then fully determined by the source transition probabilities and the topology of τ. Specifically, for every $\gamma^{(c)}$, we consider the tree τ and determine the transitions of all possible $(\gamma^{(i)}, v^{(j)})$, producing the information bits, according to $P(s_n|S_{n-1} = \gamma^{(c)})$. This is shown in figure 3.3 for the three symbol source of section 1.3.2.

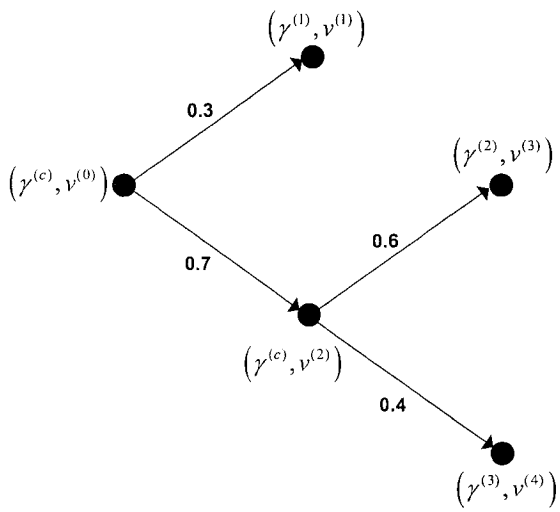


Fig. 3.3 Example of determining the transition probabilities of X. We have assumed a three symbol source with symbols corresponding to $\gamma^{(1)} =$ '1', $\gamma^{(2)} =$ '01', $\gamma^{(3)} =$ '00'. The last completed symbol is $\gamma^{(c)}$ for generality and $P(s_n|S_{n-1} = \gamma^{(c)}) = (0.3, 0.42, 0.28)$ resulting in the labeled transitions.

We note that in general, when $v^{(j)}$ is a leaf vertex, then $\gamma^{(i)}$ is necessarily the corresponding symbol. Hence, not all pairs $(\gamma^{(i)}, v^{(j)})$ are possible. In other words, we can consider that knowledge of $v^{(j)}$ is irrelevant when a new symbol terminates and we denote such states by $(\gamma^{(i)}, v^{(0)})$ where $v^{(0)}$ is the root vertex of τ. Thus, the state-space of X is reduced to $\mathcal{X} = \mathcal{S} \times \mathcal{T}$ where $\mathcal{T}$ is the set of *inner vertices* of τ. For the example of the three symbol source the state-space $\mathcal{X}$ is given by,

$$\begin{aligned} \mathcal{X} = \{&(\gamma^{(1)}, v^{(0)}), (\gamma^{(1)}, v^{(2)}), \\ &(\gamma^{(2)}, v^{(0)}), (\gamma^{(2)}, v^{(2)}), \\ &(\gamma^{(3)}, v^{(0)}), (\gamma^{(3)}, v^{(2)})\} \end{aligned} \tag{3.2}$$

We have therefore completely specified the state-space representation of the Markov source and source coder. The result is a Markov process X with the transitions from X_k to X_{k+1} producing the information bit U_k. Once again, using the natural ordering imposed by the bit clock, $X_0, X_1, U_1, X_2, U_2, \ldots, X_K, U_K$, the joint distribution factors according to,

$$P(x_0, \ldots, x_K, u_1, \ldots, u_K) = P(x_0) \prod_{k=1}^{K} P(u_k|x_{k-1}, x_k) P(x_k|x_{k-1}) \tag{3.3}$$

The corresponding Bayesian network is shown in figure 3.4. We note that the conditional prob-

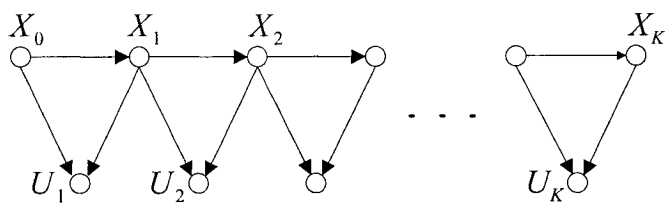


Fig. 3.4 Bit clock model for the Markov source and source coder.

abilities $P(u_k|x_{k-1}, x_k)$ and $P(x_k|x_{k-1})$ can be specified by matrices formed with the sets of corresponding probability mass functions $p(u_k|x_{k-1}, x_k)$ and $p(x_k|x_{k-1})$. The prior probabilities $P(x_0)$ may be specified by,

$$p(x_0) = \begin{cases} p_i & x_0 = (\gamma^{(i)}, v^{(0)}) \\ 0 & \text{otherwise} \end{cases}$$

where p_i is the a-prior probability of the source symbol corresponding to $\gamma^{(i)}$. As long as the total bit sequence length K is known, the topology of the Bayesian network is well defined for both CLC's and VLC's. Thus the resulting graph is generally amenable to Belief Propagation. Unfortunately, since a product Markov source and source coder model was derived, the complexity

is increased. Indeed, with $|\mathcal{T}| \simeq |\mathcal{S}|$, we have that $|\mathcal{X}| = |\mathcal{S}||\mathcal{T}| \simeq |\mathcal{S}|^2$. Hence Belief Propagation on the given Bayesian network would result in complexity of approximately $O(|\mathcal{S}|^4)$ (see previous chapter, section 2.4.2). However the complexity should not be evaluated so loosely because the transition matrix of X is in fact very sparce as each state is allowed only two possible transitions. Therefore a careful handling of the product model, should result in a complexity of $O(|\mathcal{S}|^2)$, equivalent to the complexity of the Markov source alone. More on this point later.

3.2.3 The channel coder

Deriving the bit clock Bayesian network for the channel coder is a much easier exercise. We simply rely on a state-space representation of the channel code. This directly captures the case of block codes and convolutional codes. As for any other kind of channel code, a state-space representation, if not immediately available, may always be derived. It is assumed that the channel code has X' as a state variable and with no loss in generality, we assume a bit clock recursion for the state equation with the output depending on the current state. Hence the channel coder is seen to take information bits one at a time and yields a number of redundant bits, possibly none. We denote by R_k for simplicity the *sequence* of redundant bits, $R_{k,1}, R_{k,2}, \ldots, R_{k,M}$, obtained at time k. Once again, using the natural ordering imposed by the bit clock, namely $X'_0, U_1, X'_1, R_1, U_2, X'_2, R_2, \ldots, U_K, X'_K, R_K$, the joint distribution over the random variables involved factors according to,

$$P(x'_0, \ldots, x'_K, u_1, \ldots, u_k, r_1, \ldots, r_K) = P(x'_0) \prod_{k=1}^{K} P(r_k|x'_k) P(x'_k|x'_{k-1}, u_k) P(u_k) \tag{3.4}$$

The corresponding Bayesian network is shown in figure 3.5. We have assumed for simplicity that

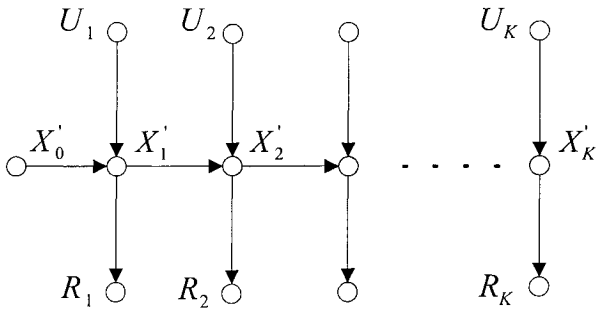


Fig. 3.5 Bit clock model for the channel coder.

a rate 1/2 systematic channel code is used. Hence each R_k corresponds to one bit as shown. If for example a rate 1/3 systematic channel code were to be used, there would be two nodes emanating

from X'_i, one for $R_{i,1}$ and another for $R_{i,2}$ and so on. The conditional probabilities $P(r_k|x'_k)$ $P(x'_k|x'_{k-1}, u_k)$ may be derived; the same is true for the prior distribution $P(x'_0)$ since we can initialize the channel coder to a known state. Again, if knowledge of the bit sequence length K is known, the topology of the graph representing the channel coder is well defined. Thus Belief Propagation is generally applicable and will result in a complexity of $O(|\mathcal{X}'|^2)$.

3.2.4 The entire coding chain

With the Bayesian network of the Markov source and source coder, and the Bayesian network of the channel coder available, deriving the graph corresponding to the entire coding chain is straightforward. The quantities involved are X, U, X', R. Choosing the natural ordering imposed by the bit clock on the random variables as $X_0, X'_0, X_1, U_1, X'_1, R_1, X_2, U_2, X'_2, R_2, \ldots, X_K, U_K, X'_K, R_K$, the joint distribution factors according to,

$$P(x_0, \ldots, x_K, u_1, \ldots, u_K, x'_0, \ldots, x'_K, r_1, \ldots, r_K) = \\ P(x_0)P(x'_0)\prod_{k=1}^{K} P(r_k|x'_k)P(x'_k|x'_{k-1}, u_k)P(u_k|x_{k-1}, x_k)P(x_k|x_{k-1}) \tag{3.5}$$

The appropriate graph is shown in figure 3.6. The graph shows the Bayesian network corre-

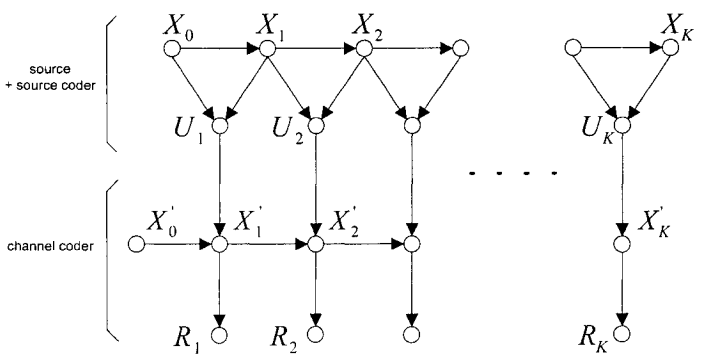


Fig. 3.6 Bit clock model for the entire coding chain.

sponding to the entire coding chain. The connections (directed links) between the variables are intuitively meaningful. Indeed, we could have essentially drawn the graph without any consideration to the factorization of the joint distribution by noting which variables a particular node depends on, not in the statistical dependence sense but rather in the sense of *causality*. Hence for example X_0 *causes* X_1, both of which *cause* U_1. The same may be said about X'_0 which along with U_1 causes X'_1, which finally causes R_1. Although this is a natural consequence of

using the ordering imposed by causal time, it nonetheless demonstrates that a Bayesian network corresponding to a particular ordering graphically reveals the conditional dependence relations of the random variables. Another relevant remark lies in the fact that the only source of *randomness* of the above Bayesian network is in the variables X_i and their inter-connections, with all other variables being deterministically related.

We reiterate the fact that the topology of the above graph is deterministic and well defined as long as the total transmitted sequence length, K is known. And the graph is therefore again amenable to Belief Propagation in that sense for both CLC's and VLC's. Since the graph represents the *serial connection* of the Markov source and source coder model with the model of the channel coder, the constituent components have been isolated and the number of required computations are $O(|\mathcal{X}|^2) + O(|\mathcal{X}'|^2)$ where the first term is with respect to the Markov source and source coder model and the last one is with respect to the channel coder model.

Finally, the assumption of an order one stationary Markov source S and that of a block length one source coder may be relaxed to higher order stationary processes and source codes with greater block length. The steps to follow in the derivation of the model for the coding chain are entirely analogous and networks with the same topology are obtained. This comes at the cost of an increase in the state-space $|\mathcal{X}|$ reflecting the increase in the coding complexity.

3.3 Joint Decoder

In light of the previous section, graphs with the same topology as the one in figure 3.6 may be built to represent any coding chain employing separate source and channel coding on a general, finite order, stationary Markov source. Therefore, such a topology may be constructed by the receiver in order to achieve a *joint decoding* scheme. The only quantities lacking for a complete specification of the graph would then be the conditional probability measures quantifying each link. However, as previously discussed, the only source of randomness in the Bayesian network of the entire coding chain is in the variables X_i with their inter-connections depending on the source transition probabilities; all other links are deterministically dependant on the source coder and the channel coder. Hence, as long as the decoder assumes knowledge of the source transition probabilities, it will have access to a fully specified graph representing the dependencies between all variables of the coding chain. This is in addition to the knowledge of the source coder, the channel coder and the length of the received bit sequence K, all of which are usually assumed in *any* deterministic decoding rule. In the following, we will assume that the decoder has access to a fully specified graph like that of figure 3.6.

3.3.1 Incorporating knowledge of the received data stream

The decoder, by definition, also has access to the received data stream. Here, we show how such knowledge may be incorporated into the graph available to the decoder. We assume that the observation on the transmitted data to be given by,

$$Y_i = a_i U_i + v_i \quad \text{and} \quad Z_i = b_i R_i + u_i \tag{3.6}$$

where $a_i, b_i \in \mathbb{R}$. We further assume that v_i, u_i are uncorrelated gaussian random variables with,

$$Ev_k v_j = \sigma_u^2 \delta_{kj} \quad \text{where } \sigma_u^2 \in \mathbb{R}, k, j \in \mathbb{Z}^+$$
$$Eu_k u_j = \sigma_r^2 \delta_{kj} \quad \text{where } \sigma_x^2 \in \mathbb{R}, k, j \in \mathbb{Z}^+$$

This captures the general case of the Rayleigh fading channel. Since observation Y_i is statistically dependent on U_i alone and since observation Z_i is also statistically dependent on R_i alone, we may simply include them as shown in figure 3.7. We note that we did not label the nodes Y_i, Z_i for simplicity. However they are depicted differently in order to emphasize the fact that their functionality, inherently different from that of remaining nodes, is as discussed in our derivation in section 2.3.3. In the case that the channel coder is not a systematic one, we may simply remove[2] the pointwise observations Y_i on the information bits. We included a constraint on symbol

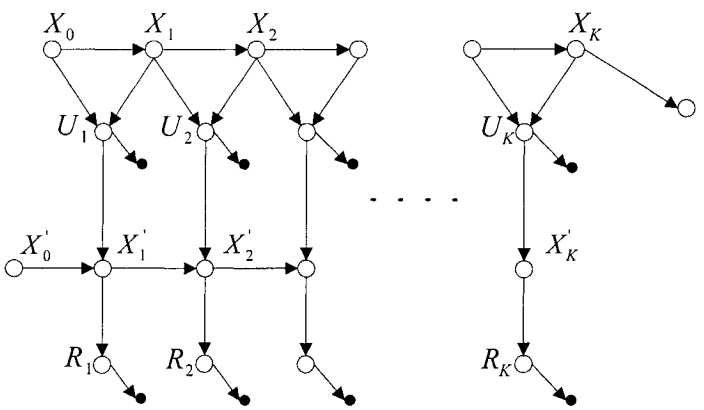


Fig. 3.7 Incorporating observation of the transmitted bit sequence.

termination that essentially ensures that the last variable $X_K = (\Gamma_K, V_K)$ indeed corresponds to

[2] from here on we will always show pointwise observations on U_i but the reader should keep in mind that they may be removed.

the end of a symbol. Thus the depicted node sends a constant message $\lambda_{term}(x_K)$ to X_K

$$\lambda_{term}(x_K) = \begin{cases} 1 & V_K = v^{(0)} \\ 0 & \text{otherwise} \end{cases}$$

This important constraint in the VLC case, allowing the synchronization of symbols both at the beginning and the end of the data stream may be removed in the CLC case. Interestingly enough, this synchronization comes for free, so to speak, and does not have to be based on RVLC's.

3.3.2 Incorporating knowledge of the transmitted symbol sequence length

Another important information, which in the VLC case may assist the joint decoder, lies in the knowledge of the transmitted symbol sequence length N. This information may be incorporated by considering the process $W = (X, C)$ that is to replace X. In the bit clock time realization of that process $W_k = (X_k, C_k) = (\Gamma_k, V_k, C_k)$, C_k represents the *number* of completed symbols at time k. The transition probabilities of (X, C) immediately follow so that including knowledge of N amounts to setting the constraint on symbol termination to deliver a constant message $\lambda_{term}(w_K)$ to W_K

$$\lambda_{term}(w_K) = \begin{cases} 1 & V_K = v^{(0)}, C_K = N \\ 0 & \text{otherwise} \end{cases}$$

Incorporating knowledge of N comes at a dramatic increase of the state-space for the Markov source and source coder model. Indeed, $|\mathcal{W}| = N \cdot |\mathcal{X}| = N \cdot |\mathcal{S}||\mathcal{T}|$. It does come with the added advantage of relaxing the earlier assumption that the source is stationary. Indeed assuming that the probabilities $P(s_n|s_{n-1})$ are *varying*, changing with the n^{th} symbol, with C_k now available, the decoder may appropriately select the source transition probabilities and accordingly determine how to quantify the links between W_k and W_{k+1}.

In the remainder of this text, we will denote W by X, in order to maintain a more uniform notation, and since the discussion to follow is applicable to both cases. When necessary, the context should make it clear to the reader which case is being treated.

3.3.3 Applying Belief Propagation on the available graph

To recapitulate, a fully specified Bayesian network is available at the decoder along with pointwise observations, that is the received data, on the redundant bit sequence R and possibly on the information bit sequence U, in the case a systematic channel code is used. We have redrawn the graph in figure 3.8 for reference. Ideally, given observations $Y = y$ and $Z = z$ and supposing Belief

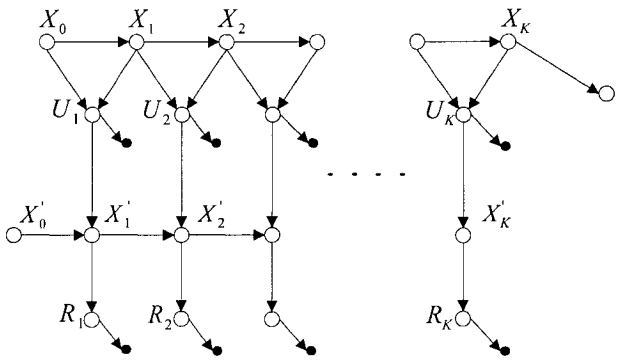


Fig. 3.8 Belief propagation for decoding.

Propagation were to converge to correct posterior marginals, we would have successfully designed a reduced complexity[3], optimal joint decoder with the belief of node U_k yielding $P(u_k|y,z)$ for the k^{th} information bit, whilst incorporating all possible sources of redundancy: the source memory, the residual redundancy of the source coder and the redundancy introduced by the channel coder. Setting the estimate $\hat{U}_k$ of U_k as,

$$\hat{U}_k = \arg\max_{u_k} P(U_k = u_k|y,z) \tag{3.7}$$

provides us with MAP estimates on the bits U_k. Estimating the symbols, would simply involve estimating the beliefs of the appropriate nodes X_k. In the CLC case, with codewords of length l, X_{nl} necessarily corresponds to S_n and hence its belief would yield $P(s_n|y,z)$. Whereas in the VLC case, one must interrogate the beliefs of all of the nodes X_k and combine the information to yield $P(s_n|y,z)$. Note however that the situation is in fact much simpler given that we seek a (hard) MAP estimate $\hat{S}_n$ of S_n,

$$\hat{S}_n = \arg\max_{s_n} P(S_n = s_n|y,z) \tag{3.8}$$

This quantity may be obtained by setting the value of each X_k to the state $\hat{X}_k$ that exhibits the highest posterior probability, that is $\hat{X}_k = \arg\max_{x_k} P(X_k = x_k|y,z)$. Since the states that correspond to a symbol termination are distinguishable in that $V_k = v^{(0)}$, an estimate $\hat{S}$ immediately follows. When knowledge of the symbol sequence length N is not incorporated, $\hat{S}$ may contain either more or less than N symbols.

Unfortunately, the situation is somewhat more bleak. Indeed, the graph is not singly connected.

[3] when compared with the optimal joint decoding of [15]

In fact, it contains a great deal of loops (undirected cycles) and thus Belief Propagation is not guaranteed to converge to *correct* posterior marginals. One solution that would render the graph singly connected is given by the aforementioned method of *node aggregation*. If we were to combine each pair X_{k-1} and X_k with X'_k into a single node, the result would be a graph that is a tree. Belief Propagation on such a graph would converge to correct quantities however this solution is equivalent to that proposed by [15] and suffers from the same intractable complexity issue. The other methods that allow Belief Propagation to converge on non-singly connected networks — conditioning and stochastic relaxation — are equally if not more computationally expensive. It appears that the Bayesian networks framework reveals the same conclusion as the discussion of the first chapter, namely that suboptimal joint decoders are required.

3.3.4 Turbo joint decoding scheme

One possible solution for a suboptimal joint decoder is inspired by the principles of serial turbo codes. Indeed, it was noted that the simple introduction of an *interleaver*, between the Markov source and source coder model and the model of the channel coder, increases the average length of the loops making short undirected cycles become long. This is shown in figure 3.9. A graphical

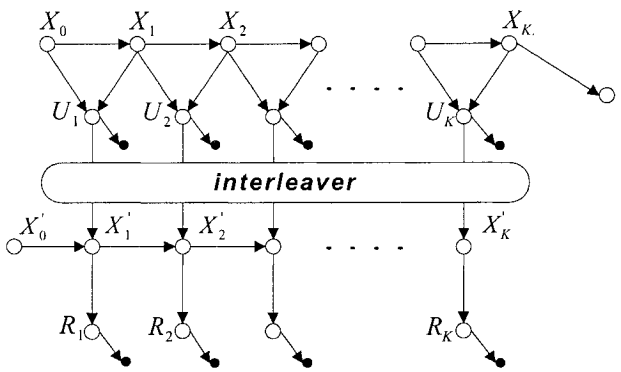
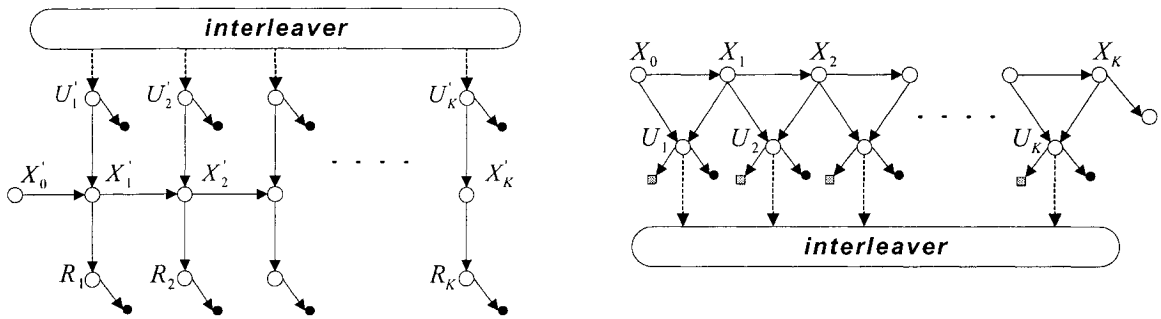


Fig. 3.9 Introduction of an interleaver to increase the average length of the loops.

model containing undirected cycles with a large average length may be locally approximated by a singly connected network. This takes into account the fact that the correlation between the nodes is likely to decay exponentially fast with distance. Hence, Belief Propagation may be applied on the graph to yield good approximations to the *correct* posterior marginals. In agreement with the traditional architecture of turbo algorithms, an iterative scheme is designed that alternates the use of the channel coder model and the joint Markov source and source coder model. Specifically the graph is divided into two subgraphs with the information bit sequence U reproduced as shown in figure 3.10. Figure 3.10(a) shows the first step of the first iteration. U'_k is used to denote the



(a) channel coder subgraph.

(b) Markov source and source coder subgraph.

Fig. 3.10 Iterative scheme

interleaved version of the information bits U_k. For this particular subgraph, a-prior distributions $P^0(u_i)$ are required. An i.i.d equiprobable distribution is assumed. Belief Propagation, on the channel coder subgraph, is carried out in two passes at the end of which an estimates $P^0(u_k|y,z)$ are obtained. Following standard extrinsic information computations, the following quantity is defined,

$$Ext^0_{U_k}(z|y) = \frac{P^0(u_k|y,z)}{P(u_k|y_k)} \tag{3.9}$$

representing the remaining information, regarding U_k, carried by Z once Y_k is known. At the second step of the first iteration, the $Ext^0_{u_k}(z|y)$ quantities are passed, to the Markov source and source coder subgraph (figure 3.10(b)), as pointwise measurements or observations on the U_k's. They are depicted as gray squares. Once again, Belief Propagation is carried out in two passes yielding estimates $P^1(u_k|y,z)$. This closes the loop of the first iteration. As for the new prior $P^2(u_k)$ to be sent to the channel coder as the first step of the second iteration, the quantity used is,

$$P^2(u_k) = P(u_k)\frac{P^1(u_k|y,z)}{P(u_k|y_k)Ext^0_{U_k}(z|y)} \tag{3.10}$$

The second iteration is carried out by repeating the same steps.

We note that, in terms of the larger context of joint decoding, this solution represents a sub-optimal, limited complexity, joint decoding scheme taking into account three types of redundancies present in the coding chain: the source memory, the source coder's residual redundancy as well the redundancy introduced by the channel coder.

Chapter 4

An Enhanced Joint Source Channel Decoder: Theory and Results

In this chapter we present, analyze and discuss a new joint source channel decoding scheme. The proposed algorithm, which relies on the principles of iterative decoding, is largely inspired from the developments of the previous section and takes into account three types of information: the source memory, the residual redundancy of the source coder as well as the redundancy introduced by the channel coder. We begin this chapter by presenting our joint decoding scheme. Specifically, we derive an equivalent Bayesian network representation of the coding chain and demonstrate that iterative decoding may be carried out on the resulting graph via a specific ordering of node activation. Next, we show how the proposed equivalent graph has superior convergence properties, as it not only relaxes a stringent statistical independence assumption imposed by the graph in [20] but it also contains far less undirected cycles. This is followed by an analysis which shows that our proposed scheme is in fact drastically reduced in computational complexity. Finally, computer simulations results are presented which substantiate our predicted performance gains.

4.1 Proposed Algorithm

Here, we present a new iterative joint source-channel decoding algorithm. The algorithm's novelty is based, on the one hand, on an *equivalent* Bayesian network representation of the coding chain and on the other hand, on a different approach with regards to to extrinsic information computations as well as the method of iteration.

4.1.1 An equivalent Bayesian network representation of the coding chain

In the following, we derive an equivalent Bayesian network representation of the coding chain, a representation based on a simple yet potent observation.

Recall the derivation in section 3.3.3 of the product Markov source and source coder model. The resulting state-space was originally given by $\mathcal{X} = \mathcal{S} \times \mathcal{V}$ where $\mathcal{S}$ is the state-space of the source — in other words, the set of all possible source symbols — and $\mathcal{V}$ is the set of the vertices of the binary tree τ representing the source coder's mapping. The state variable X is then specified by the pair $X = (\Gamma, V)$, where Γ, with instance $\gamma^{(i)}$, is a variable representing the last completed symbol and V, with instance $v^{(j)}$, is a variable representing the current vertex of τ. Hence, the state-space representation provided states $(\gamma^{(i)}, v^{(j)})$ whose transitions, dictated by the source transition probabilities $P(s_n|s_{n-1})$ and the topology of τ, yield the information bits. It was further noted that when $v^{(j)}$ is a leaf node, $\gamma^{(i)}$ must necessarily be the corresponding symbol. This essentially implies that when $v^{(j)}$ is a leaf vertex, it should be substituted with $v^{(0)}$, which denotes the *root* vertex of τ. Hence the state-space of X was reduced to $\mathcal{X} = \mathcal{S} \times \mathcal{T}$ where $\mathcal{T}$ is the set of *inner-vertices* of τ.

The point of interest here is that the dependence of the information bits on the transitions of states $(\gamma^{(i)}, v^{(j)})$ is a mere formality. Indeed, since the transitions of states is dictated by the topology τ of a *binary* tree, each state is allowed only two possible transitions out. More to the point, all of the transitions *into* a state $(\gamma^{(i)}, v^{(j)})$ produce the *same* output. This is immediately obvious for the case when $v^{(c)}$ is any inner vertex that is not the root vertex $v^{(0)}$, since in fact only *one* transition is allowed to any state $(\gamma^{(i)}, v^{(c)})$, namely, from the state $(\gamma^{(i)}, v^{(c')})$, where $v^{(c')}$ is the inner-vertex connected to $v^{(c)}$. As for the case of the root vertex $v^{(0)}$, there are $\mathcal{S}$ transitions into a state $(\gamma^{(i)}, v^{(0)})$, all of which indicate the *completion* of the symbol corresponding to $\gamma^{(i)}$. Hence they necessarily produce the same output, namely the last codeword bit of the source symbol corresponding to $\gamma^{(i)}$. In figure 4.1, we show for the sake of clarity, all of the allowable transitions for our previous example of the Markov source of three symbols of section 1.3.2.

It is clear that since all possible transitions into a given state $(\gamma^{(i)}, v^{(j)})$ result in the same output bit, one can equivalently consider the output bit to be a function of that given state and not of the transition. In essence, we have a Markov process X entirely identical to that of Guyader et al.[20] with respect to its state-space, its state transition probabilities $P(x_k|x_{k-1})$, but with the fundamental difference that the information bit U_k depends on X_k alone.

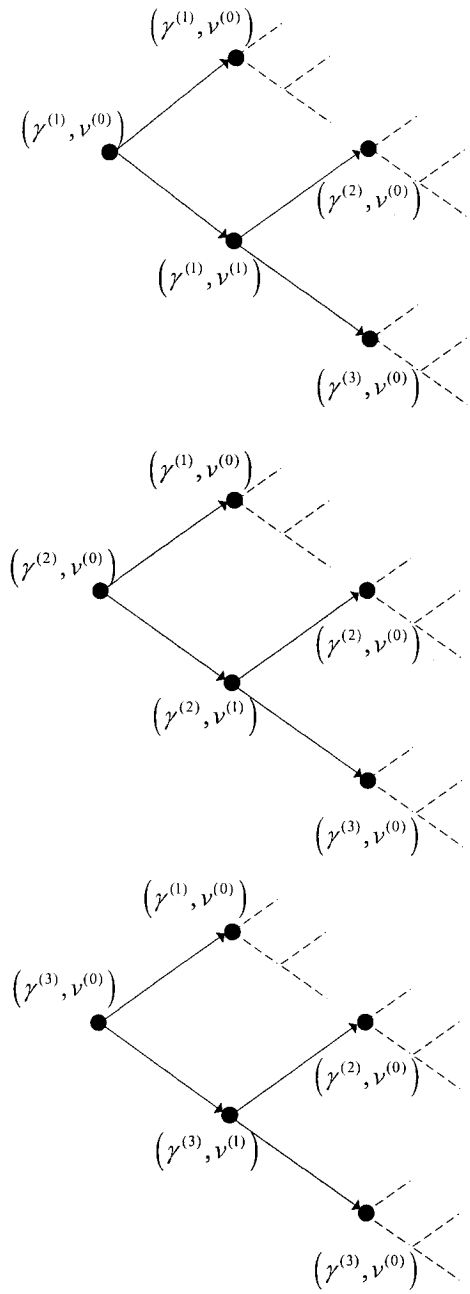


Fig. 4.1 All allowable transitions for the three symbol source in terms of the topology of the binary tree of symbols τ. From the top to bottom, the first tree assumes the last completed symbol to be $\gamma^{(1)}$, the second assumes the last completed symbol to be $\gamma^{(2)}$, whilst the third assumes $\gamma^{(3)}$ to be the last completed symbol

The corresponding *re-factored* Bayesian network for the Markov source and source coder model is shown in figure 4.2. Note that this Bayesian graph, equivalent to the graph shown in figure

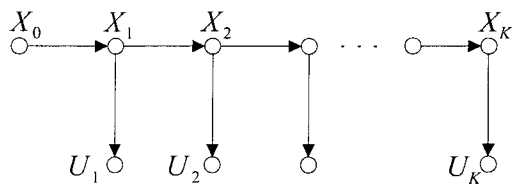


Fig. 4.2 New Bayesian network representation for the Markov source and source coder model equivalent to the graph shown if figure 3.4.

3.4, is as well valid in general for *any* order-one Markov source of symbols and *any* block length one binary source coder. Once again, as long as the total bit length K is known, the topology of the graph is entirely deterministic and includes both CLC's and VLC's. In fact all of the previous developments of the previous chapter hold with the only difference that the conditional probabilities $P(u_k|x_{k-1}, x_k)$ are now replaced with the conditional probabilities $P(u_k|x_k)$ that are available through the set of corresponding probability mass functions. In terms of the joint distribution on the variables X_i and U_i, the graph represents the following factorization,

$$P(x_0, \ldots, x_K, u_1, \ldots, u_K) = P(x_0) \prod_{k=1}^{K} P(u_k|x_k) P(x_k|x_{k-1}) \tag{4.1}$$

As for higher order Markov sources and source coders with larger block length, graphs with the same topology as that of figure 4.2 may be derived[1].

It is convenient at this point to consider the larger context that enables one to move from a state-space representation with the output depending on the transition of states to one where the output depends solely on the given state. Indeed, it is generally always possible to move from one type of representation to the other. This may be done one state at a time, by considering all the possible transitions *into* the state. Supposing there is a total number q of possible outputs resulting for those transitions, we split our state into q new states, one for each possible output. We can now consider each of the new q states to be associated with one of the outputs. This is shown in figure 4.3. The transitions *into* our original state are redirected to one of the new corresponding states. The transitions *out* of our original state are reproduced for each of the q states. This procedure is then repeated for all remaining states. We note that in general such a

[1] see section 3.2.4

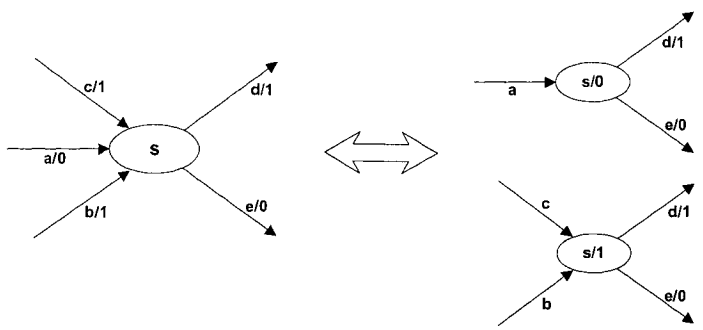


Fig. 4.3 Transforming a state-space representation with the output depending on the transition of states to one where the output depends on the given state. Letters a,b,c,d,e are used to generically denote probabilities, whereas the outputs, here chosen to be binary, are denoted by /0 or /1.

transformation will result in an increase of the state-space by as much as a factor of q. Indeed, $|\mathcal{S}_{given}| \leq q|\mathcal{S}_{trans}|$ where $\mathcal{S}_{given}$ is the state-space of a representation with outputs depending on the given state and $\mathcal{S}_{given}$ is the state-space of the corresponding representation with outputs depending on the transition of states. However when all of the transitions into a state yield the same output, as is the case for the Markov source and source coder model, such a transformation comes with no cost: $|\mathcal{S}_{given}| = |\mathcal{S}_{trans}|$.

With a new bit-clock equivalent Bayesian network for the Markov source and source coder model, we may now derive an equivalent Bayesian network representing the entire coding chain. The channel coder is not changed so that the result is a serial concatenation of the two model as shown in figure 4.4.

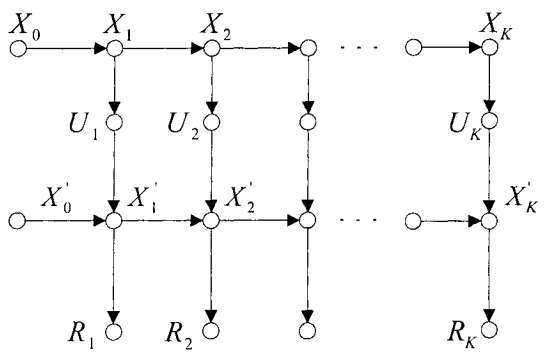


Fig. 4.4 New bit clock Bayesian network representation for entire coding chain equivalent to graph shown in figure 3.6.

The graph represents the following factoring of the joint distribution,

$$P(x_0,\ldots,x_K,u_1,\ldots,u_K,x'_0,\ldots,x'_K,r_1,\ldots,r_K) = P(x_0)P(x'_0)\prod_{k=1}^{K} P(r_k|x'_k)P(x'_k|x'_{k-1},u_k)P(u_k|x_k)P(x_k|x_{k-1}) \tag{4.2}$$

Again, given knowledge of the received bit length sequence K, the topology of the graph is deterministic for both CLC's and VLC's. Finally, such a graph may be built to represent any coding chain employing separate source and channel coding on general finite-order Markov sources and comes with no cost with respect to the state-spaces of the variables involved.

4.1.2 Towards a fully consistent solution for turbo joint decoding

As discussed in the previous section, if we assume that the decoder has knowledge of the source transition probabilities $P(s_n|s_{n-1})$ that it has available to it a fully specified graph as the one in figure 4.4. We may incorporate knowledge of the received data stream and Belief Propagation may be applied in order to yield MAP estimates on the information bits U_k and on the symbols S_n. Unfortunately, once again, we are left with a non-singly connected graph, one that in facts contains a significant number of loops (undirected cycles). The turbo decoding solution, as proposed by Guyader et al. [20], presents itself as a viable alternative that would yield good approximations to the posterior marginal probabilities $P(x_k|y,z)$ and $P(u_k|y,z)$. Indeed, we may insert an interleaver, just as described earlier, between the Markov source and source coder model and the model of the channel coder. This again increases the average length of the undirected cycles, so that the graph may be better locally approximated by a singly connected network. This is shown in figure 4.5 where we have as well included the pointwise observations Y_i on U_i and Z_i on R_i as well as the constraint on symbol termination.

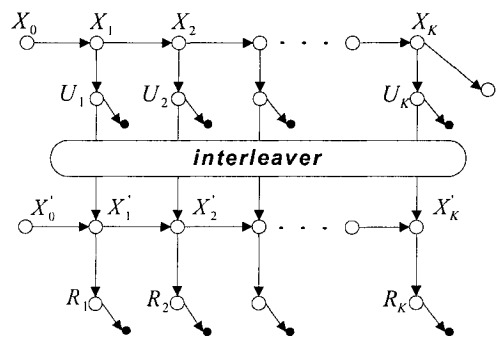


Fig. 4.5 Inserting an interleaver for the new Bayesian network available to the decoder.

Iterative decoding may then be applied by alternating the successive use of the channel coder model followed by the Markov source and source coder model as discussed in section 3.3.4. However we propose a different approach for the iterative scheme. Specifically, we do no separate the graph into two subgraphs and compute the extrinsic information quantities *externally* to the belief propagation process. Rather we leave the graph *connected yet interleaved* and we achieve iterative decoding via a specific ordering of node activation.

In particular, we start by assuming that the continuous observation nodes Y_i and Z_i are already activated so that they are readily providing the $\lambda_{Y_i}(u_i)$ and $\lambda_{Z_i}(r_i)$ messages to U_i and R_i. Note that we can also assume that all of the R_i nodes are already activated and will not be activated at any later time. This is done to simplify the iterative scheme and comes with no loss in generality since the only *useful* information that node R_i provides to the rest of the graph is through the message $\lambda_{R_i}(x'_i)$, which is not affected by the incoming message to R_i. Hence the $\lambda_{R_i}(x'_i)$ are in fact always constant. All other messages are initialized with equal weight on every coordinate. In terms of the information bits, this results in equiprobable initial probabilities $P^0(u_k)$. The iterative scheme then consists of activating nodes U_1 to U_k, performing Belief Propagation in two passes on the X'_k nodes, activating nodes U_1 to U_k again and finally perform two passes on the X_k nodes. This closes the loop of the first iteration and the process is repeated for the subsequent iterations. The 1^{st} iteration is shown in Table 4.1 below. We have used y_1^k to denote the vector $y_1, \ldots, y_k$.

Table 4.1 Iterative scheme as a particular ordering of node activation: 1^{st} iteration

Node Activated (Channel coder)	$BEL(\cdot)$	Node Activated (Markov source & source coder)	$BEL(\cdot)$
U_1		U_1	
$\vdots$	$P^0(u_k\|y_k)$	$\vdots$	$P^0(u_k\|y,z)$
U_K		U_K	
X'_0		X_0	
$\vdots$	$P^0(x'_k\|y_1^k,z_1^k)$	$\vdots$	$P^0(x_k\|y_1^k,z_1^k)$
X'_K		X_K	
$\vdots$	$P^0(x'_k\|y,z)$	$\vdots$	$P^0(X_k\|y,z)$
X'_0		X_0	

As the second iteration begins, the activation of nodes U_1 to U_K yields $P^1(u_k|y,z)$ and we continue the process. Note that if we wish to read out the MAP estimate on the bit sequence U at the *end* of the first iteration, we must use $P^1(u_k|y,z)$ and hence activate the nodes U_k one more time. The same applies to the subsequent iterations.

A justification with respect to our iterative scheme is in order. First, we recall that since the graph is interleaved, node U_k is in general no longer connected to node X'_k but to say node X'_l. This is shown in figure 4.6. Let us consider the first iteration as node U_k updates its belief in the

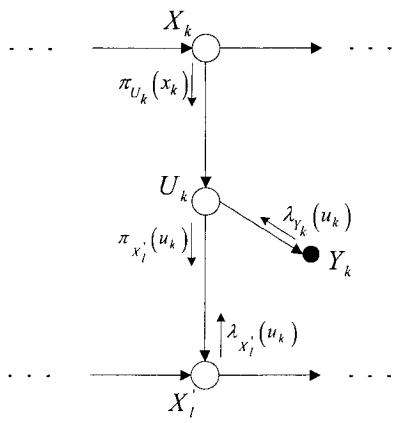


Fig. 4.6 Extrinsic information as Belief Propagation messages.

first set of node activations for the Markov source and source coder model. Its belief is given by,

$$BEL(u_k) = \alpha \lambda_{X'_l}(u_k) \lambda_{Y_k}(u_k) \sum_{x_k} P(u_k|x_k) \pi_{U_k}(x_k) \tag{4.3}$$

Rewriting the equation in terms of the quantities defined in table 4.1, we have,

$$\begin{aligned} BEL(u_k) &\triangleq P^0(u_k|y,z) \\ &= \alpha_1 \lambda_{X'_l}(u_k) \lambda_{Y_k}(u_k) P^0(u_k) \\ &= \alpha_2 \lambda_{X'_l}(u_k) P^0(u_k|y_k) \end{aligned} \tag{4.4}$$

where the first equality follows from the law of total probability and since message $\pi_{U_k}(x_k)$ was initialized with equal weight on each coordinate. The second equality immediately follows from Bayes' law. Hence, the message $\lambda_{X'_l}(u_k)$ which node U_k receives from X'_l, is in fact given by, after normalization,

$$\lambda_{X'_l}(u_k) = \frac{P^0(u_k|y,z)}{P^0(u_k|y_k)} \tag{4.5}$$

and is therefore *equal* to $Ext^0_{U_k}(Z|Y)$ defined in the previous chapter which represents the standard quantity used in iterative schemes: that is the remaining information, regarding U_k, carried by Z once Y_k is known. As for the first step of the second iteration, when node U_k updates its belief it sends a message $\pi_{X'_l}(u_k)$ to X'_l. At this point message $\lambda_{X'_l}(u_k)$ is unchanged and U_k has received

all its additional information from X_k. Hence message $\pi_{X_l'}(u_k)$ is simply given by,

$$\pi_{X_l'}(u_k) = \frac{P^1(u_k|y,z)}{\lambda_{X_l'}(u_k)} \tag{4.6}$$

Note that in [20], the quantity defined is in fact, $P(u_k)\pi_{X_l'}(u_k)$, however it is clear that $P(u_k)$ is already contained in the additional information that X_k sends to U_k and as such, it is part of $P^1(u_k|y,z)$.

The intuition behind our scheme reduces to the fact that the messages on the link $U_k \rightarrow X_l'$ contain *disjoint* information and as such, should be used as the appropriate extrinsic information quanities. Our proposed scheme comes with the added advantage of forgoing both the additional overhead of separating the graph into two subgraphs with the bit sequence U reproduced, as well as the overhead required in computing the extrinsic information quantities externally to the Belief Propagation process. Our proposed scheme also shows that a particular organization of node activation in Belief Propagation immediately results in a turbo joint source-channel decoding algorithm and is in agreement with the developments of McEliece [32]. Hence it represents a fully consistent solution to the iterative (turbo) joint source-channel decoding problem within the Bayesian networks framework.

4.2 Theoretical Discussion on the Proposed Algorithm

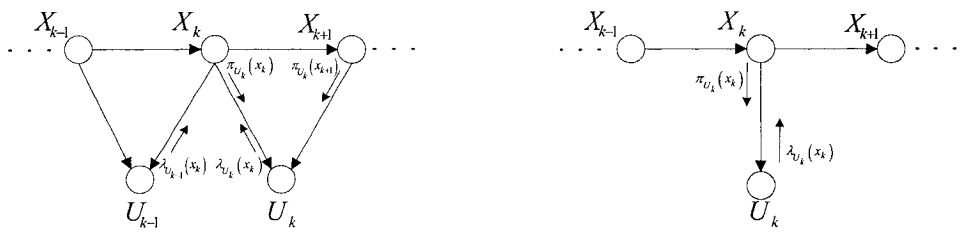
In this section, we discuss and analyze some of the properties of our proposed algorithm. We begin by giving exposing the improved convergence properties that essentially follow from the new graph itself. Next we present a complexity analysis that demonstrates that our algorithm has a significantly reduced computational complexity.

4.2.1 Improved Convergence Properties

Relaxing a stringent assumption

The new equivalent graph relaxes a relatively stringent assumption. Consider figure 4.7(a) representing Guyader et al's Markov source and source coder model. Indeed, when node U_k updates its belief, it assumes that the information from X_k and X_{k-1}, in the form of messages $\pi_{X_k}(u_k)$ and $\pi_{X_{k+1}}(u_k)$, is statistically independent. This is clearly not the case since there exists a link between X_k and X_{k+1}. The same may be said about node X_k which assumes the information from U_{k-1} and U_k to also be statistically independent. Thus, Belief Propagation on such a graph will generally result in incorrect posterior probabilities for all the nodes in the Markov source and

source coder model. Moreover, Belief Propagation in two passes will generally not result in graph equilibrium. Essentially the problem lies in the fact that the graph contains undirected cycles.



(a) Guyader et al.'s [20] graph (b) Our proposed graph

Fig. 4.7 Messages in the Markov source and source coder model

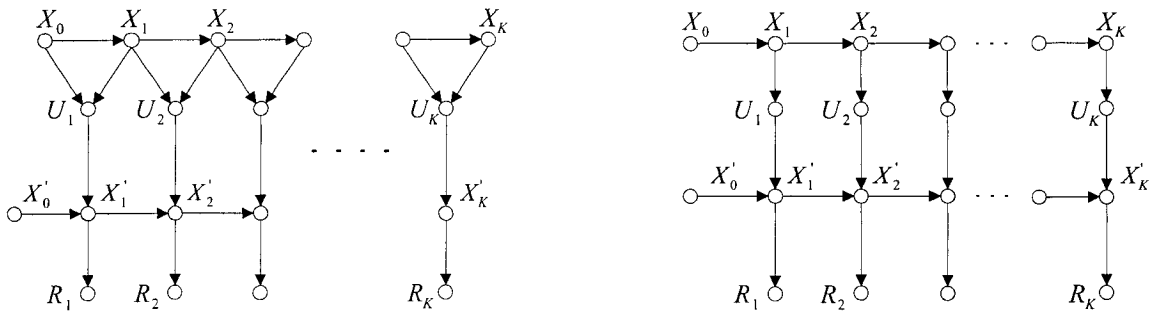
On the other hand, our equivalent graph is singly connected and is in fact a tree. Hence belief propagation in two passes is guaranteed to converge to correct posterior probabilities and a state of equilibrium will be reached. In terms of iterative decoding, the aforementioned problem in Guyader et al's Markov source and source coder model is still present and the corresponding sub-graph will not yield correct posterior probabilities according to the observations $Y = y$ and pointwise extrinsic information measurements $Ext^0_{U_k}(Z|Y)$. However, our equivalent graph solves this problem with all nodes reaching their correct posterior probabilities according to the observations $Y = y$ and the $\lambda_{X'_l}(u_k) = Ext^0_{U_k}(Z|Y)$ messages representing the extrinsic information. Hence we expect a better performance for our algorithm in decoding.

Reduction in the number of undirected cycles

The proposed algorithm comes with the added advantage that our equivalent representation of the entire coding chain contains significantly fewer undirected cycles (loops). This is immediately apparent when one compares the two graphs as shown in figure 4.8 since all of the loops $\{X_{k-1}, X_k, U_k, X_{k-1}\}$ have been effectively removed. However, a by-product of the removal of these loops is the elimination of many other loops in the overall graph. Such is the case of all the loops $\{X_k, U_k, X'_k, X'_{k+1}, U'_{k+1}, X_k\}$ for example. The number of loops for the graph in figure 4.8(a), assuming a bit sequence of length K, can be computed[2] to be,

$$2^{K+2} - 3K - 4 \tag{4.7}$$

[2]we do not show the derivation because they bare little pertinence to the overall understanding



(a) Guyader et al.'s [20] graph

(b) Our proposed graph

Fig. 4.8 Comparing the number of undirected cycles in the overall graphs

This exponentially growing number of loops stands in sharp contrast with the number of loops for our proposed graph in figure 4.8(b), which was found[3] to be,

$$\frac{1}{2}K(K-1) \tag{4.8}$$

Figure 4.9 shows the number of loops in our equivalent graph as a percentage of the number of undirected cycles in the graph of Guyader et al. [20]. This drastic reduction in the number of

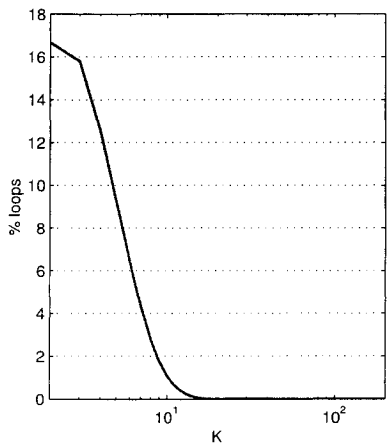


Fig. 4.9 Number of loops in our equivalent graph as a percentage of the number of loops in the original graph [20] for different lengths, K, of source sequences in bits.

loops necessarily implies that our graph may be better locally approximated by a singly connected graph. This is in turn implies a better approximation of the posterior probabilities of each node and thus a better performance in the context of decoding.

[3] we do not show the derivation because they bare little pertinence to the overall understanding

4.2.2 Computational Complexity Reduction

In this section we show that our proposed algorithm is significantly reduced in computational complexity. First we consider the case when a black box[4] belief propagation is applied on the graph available to the decoder. In such a case, it is clear from our previous discussion in section 2.4.2, that both algorithms have an order of complexity $O(|\mathcal{X}|^2)+O(|\mathcal{X}'|^2)$. For this reason we must consider instead the number of required operations — multiplications and additions. However, as was noted earlier, this case is not very realistic since the conditional probability matrices that quantify the links in the graph are very sparse. To this end, we also evaluate the reduction in complexity of our proposed algorithm when an efficient implementation of Belief Propagation is applied. The number of operations per node activation in both these cases is examined and derived in detail in Appendix A, which also contains an analysis with respect to the sparse nature of the conditional probability matrices. The reader is encouraged to refer to Appendix A for a more lucid reading of this section.

Black box implementation

The number of operations for the different nodes in our graph is shown in table 4.2 and were determined from equation A.10. We note that only the second order terms were kept for each node. We also note that we did not consider the operations necessary in the activation of nodes Y_k, Z_k and R_k since they need only be activated once and more importantly, these operations are negligible with respect to the overall complexity. All of these assumptions can be shown to yield approximations, in the comparison of complexities, accurate to 1 percentage point.

Table 4.2 Comparison of operations — multiplications and additions — per node activation of the proposed algorithm and Guyader et al.'s [20]. Black box case.

	Proposed scheme		Guyader et al.'s scheme	
NodeActivated	Multiplications	Additions	Multiplications	Additions
X_k	$2\|\mathcal{X}\|^2$	$2\|\mathcal{X}\|^2$	$2\|\mathcal{X}\|^2$	$2\|\mathcal{X}\|^2$
U_k	–	–	$8\|\mathcal{X}\|^2$	$6\|\mathcal{X}\|^2$
X'_k	$10\|\mathcal{X}'\|^2$	$6\|\mathcal{X}'\|^2$	$10\|\mathcal{X}'\|^2$	$6\|\mathcal{X}'\|^2$

The reduction in computational complexity comes mainly from the fact that since node U_k is no longer connected to X_{k-1} and X_k but rather to X_k alone, the operations it performs are now proportional to $|\mathcal{X}|$ instead of the $|\mathcal{X}|^2$ exhibited by Guyader et al.'s [20] algorithm. If we were to compare the complexities of the Markov source and source coder models, we note that Belief

[4]here, we mean that all messages are to be computed as in equations 2.16, 2.17 and 2.18, regardless of any particular structure of the underlying pmf's (see Appendix A).

Propagation in two passes results in two activations of node X_k and two activations of node U_k. Hence we have that the number of multiplications performed by our algorithm as a fraction of the number of multiplications performed by [20] is,

$$\frac{2|\mathcal{X}|^2}{2|\mathcal{X}|^2 + 8|\mathcal{X}|^2} = \frac{2|\mathcal{X}|^2}{10|\mathcal{X}|^2} = 20\% \tag{4.9}$$

whereas the fraction of additions performed by our algorithm is,

$$\frac{2|\mathcal{X}|^2}{2|\mathcal{X}|^2 + 6|\mathcal{X}|^2} = \frac{2|\mathcal{X}|^2}{8|\mathcal{X}|^2} = 25\% \tag{4.10}$$

Thus, our equivalent Markov source and source coder model comes with the added advantage of reducing by 80% the performed multiplications and 75% the performed additions. As for the overall decoder, its complexity is dictated by $|\mathcal{X}'|$ and by $|\mathcal{X}|$. In figure 4.10, we consider the case where the joint decoder has no knowledge of the length of the transmitted symbol sequence N, hence $|\mathcal{X}| = |\mathcal{S}||\mathcal{T}|$. We assumed further that $|\mathcal{T}| \cong |\mathcal{S}|$, a very good approximation for all binary source coders[5], and hence we have $|\mathcal{X}| \cong |\mathcal{S}|^2$. Finally, in figure 4.10, we assumed that the channel coder has 5 bits of memory so that $|\mathcal{X}'| = 2^5$ and $|\mathcal{X}|$ was varied by varying the source alphabet cardinality $|\mathcal{S}|$. The decrease in the curves is due to the fact that the channel coders's

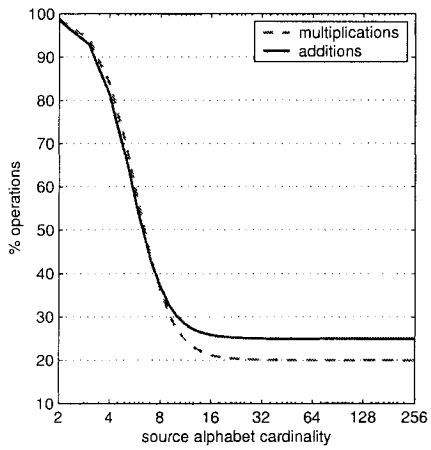


Fig. 4.10 Number of computations in our equivalent graph as a percentage of the number of computations in the original graph [20] versus the source alphabet cardinality. The joint decoder has no knowledge of N.

complexity is becoming negligible with respect to the complexity of the Markov source and source coder so that the asymptotic reduction in operations is that reduction of the Markov source and source coder model. Note that a 16-symbol source already yields the asymptotic reduction. When

[5]generally, one will find one inner-vertex for one leaf vertex

knowledge of N is incorporated, we have $|\mathcal{X}| \cong N|\mathcal{S}|^2$. In such a scenario, for any value of $N \geq 20$, the channel coder's complexity becomes negligible at a source alphabet cardinality of 4. Hence we can consider that the asymptotic reduction in computations is always obtained.

Efficient implementation

The matrices involved in the Bayesian network representation of the coding chain are very sparse. Therefore, a black box comparison of complexity is not very realistic and we should instead consider an efficient implementation comparison of complexities. In Appendix A, we show the detailed derivations of the number of operations required per node activation assuming the conditional probability matrix quantifying the node's links to its parents contains ζ non-zero elements. Appendix A also establishes upper bounds on the number of non-zero elements for all the matrices involved in the coding chain, shown here for reference.

$$\zeta_{P(X_k|X_{k-1})} = 2|\mathcal{X}| \tag{4.11}$$

$$\zeta_{P(U_k|X_k)} = |\mathcal{X}| \tag{4.12}$$

$$\zeta_{P(U_k|X_{k-1},X_k)} = 2|\mathcal{X}| \tag{4.13}$$

$$\zeta_{P(X'_k|X_k,U_k)} = 2|\mathcal{X}'| \tag{4.14}$$

where ζ_A denotes the number of non-zero entries in matrix A. Using these upper bounds and equation A.20, we can determine upper bounds on the operations that each node needs to perform for both the proposed algorithm and the algorithm in [20]. These are shown in table 4.3. Note that

Table 4.3 Comparison of operations per node activation of the proposed algorithm and [20]. Upper bounds on operations for an efficient implementation

	Proposed scheme		Guyader et al.'s scheme	
NodeActivated	Multiplications	Additions	Multiplications	Additions
X_k	$12\|\mathcal{X}\|$	$8\|\mathcal{X}\|$	$18\|\mathcal{X}\|$	$10\|\mathcal{X}\|$
U_k	$3\|\mathcal{X}\|$	$3\|\mathcal{X}\|$	$14\|\mathcal{X}\|$	$8\|\mathcal{X}\|$
X'_k	$16\|\mathcal{X}'\|$	$9\|\mathcal{X}'\|$	$16\|\mathcal{X}'\|$	$9\|\mathcal{X}'\|$

we did not include the operations for nodes Y_k, R_k and Z_k because they need only be activated once and therefore the contribution of their operations to the overall complexity is negligible. Note also, some approximations were made with respect to equation A.20 with some terms ignored. However these approximations can all be shown to yield comparisons in complexities accurate

to 1 percentage point. The efficient implementation brings the order of complexity down to $O(|\mathcal{X}|) + O(|\mathcal{X}'|)$. If we were to compare the complexities of the Markov source and source coder models, we note that Belief Propagation in two passes results in two activations of node X_k and two activations of node U_k. Hence we have that the fraction of multiplications performed is,

$$\frac{12|\mathcal{X}| + 3|\mathcal{X}|}{18|\mathcal{X}| + 14|\mathcal{X}|} = \frac{15|\mathcal{X}|}{32|\mathcal{X}|} = 47\% \tag{4.15}$$

whereas the fraction of performed additions is given by,

$$\frac{8|\mathcal{X}| + 3|\mathcal{X}|}{10|\mathcal{X}| + 8|\mathcal{X}|} = \frac{11|\mathcal{X}|}{18|\mathcal{X}|} = 61\% \tag{4.16}$$

Hence when using an efficient implementation, our Markov source and source coder model is still significantly less complex with a reduction of 53% for the multiplication operations and a reduction of 39% in the addition operations. Figure 4.11 shows the percentage operations that our overall decoder performs with respect to the overall decoder in [20] for the case that the decoder has no knowledge of N. Again, it was assumed that the channel coder has 5 bits of memory so that $|\mathcal{X}'| = 2^5$ and $|\mathcal{X}| \cong |\mathcal{S}|^2$ was varied by varying $|\mathcal{S}|$. We note that in the case of the efficient

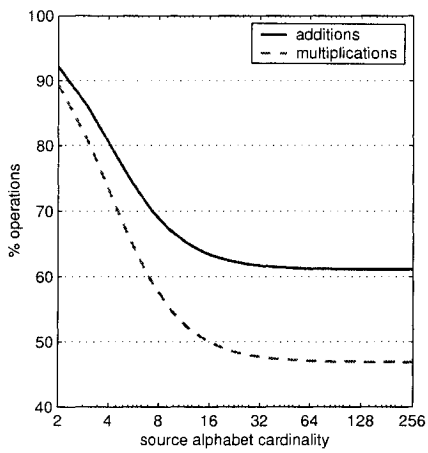


Fig. 4.11 Number of computations in our equivalent graph as a percentage of the number of computations in the original graph [20] versus the source alphabet cardinality.

implementation, the channel coder's complexity becomes negligible at a 32-symbol source and the overall decoder then yields the reduction in complexity of the Markov source and source coder model. As for the case where knowledge of N is included so that $|\mathcal{X}| = N|\mathcal{S}|^2$, we have the same situation as in the black box case. In particular, for any value of $N \geq 20$, the channel coder's complexity is negligible for a 4-symbol source and hence we can consider that the asymptotic reduction in complexity is always achieved.

4.3 Comparative Study: Results and Discussion

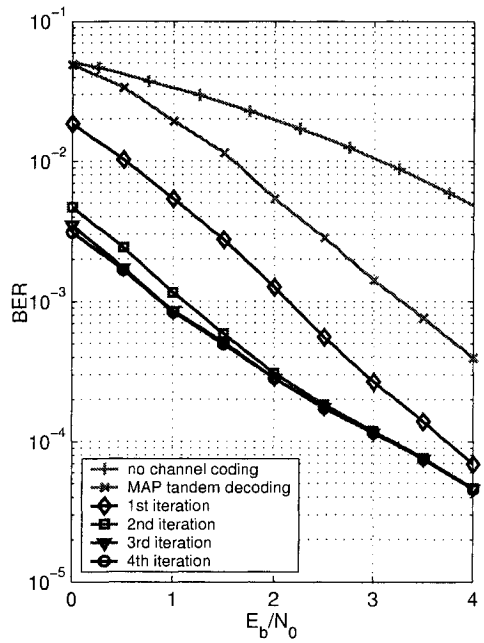
In this section, we evaluate and compare the performance of the proposed iterative scheme with the existing one in [20]. In order to do so computer simulations were carried out, implementing both these schemes. On the transmission end, the system consists of a Markov, order one, source of symbols, followed by a block length one Huffman source coder, followed by a recursive systematic convolutional channel code: the last two components were separated by a variable length interleaver. Specifically, the source used is of the Gauss-Markov variety with zero-mean and unit-variance and a correlation factor of 0.9. The source was quantized using a 3-bit uniform quantizer on the interval in order to generate discrete finite alphabet symbols. The Huffman encoder was designed according to the source statistics and yields an expected length of 2.54 bits per source symbol. The recursive systematic channel code is derived from a mother code of rate 1/2 defined by the polynomials $F(D) = 1 + D + D^2 + D^4$ and $G(D) = 1 + D^3 + D^4$. The code was augmented to a rate 3/4 by an appropriate puncturing of the redundant bit stream R_k. The variable length interleaver was based on a mother interleaver which was randomly generated by ordering a sequence of uniform random numbers. The channel was assumed to be AWGN and a binary phase shift keying (BPSK) modulation was employed.

In all of the figures to follow, we plotted bit error rate (BER) and symbol error rate (SER) for different E_b/N_0 with E_b representing the *coded* bit energy. The first curve in all of the figures corresponds to the case where no channel coding is employed: the received bit stream is therefore hard decoded assuming independent bits, to obtain the BER, followed by a hard Huffman decoding to obtain the SER. The second curve represents the commonly used tandem decoding, namely MAP channel decoding assuming an input of independent bits, followed by hard Huffman decoding. The subsequent curves show the first to fourth iterations of either the proposed iterative scheme or that in [20] which was implemented verbatim. We have organized the results into two sections for clarity. The first to follow shows the case where blocks of 50 symbols were decoded at a time and the second is the case where blocks of 200 symbols were decoded at a time.

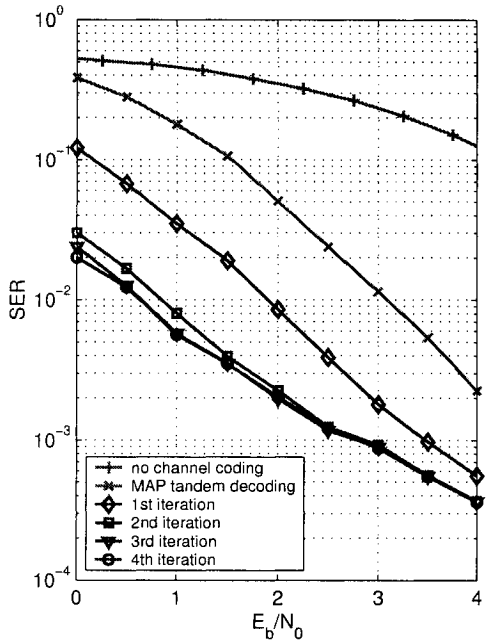
4.3.1 Blocks of 50 symbols

Joint Decoder has no knowledge of N

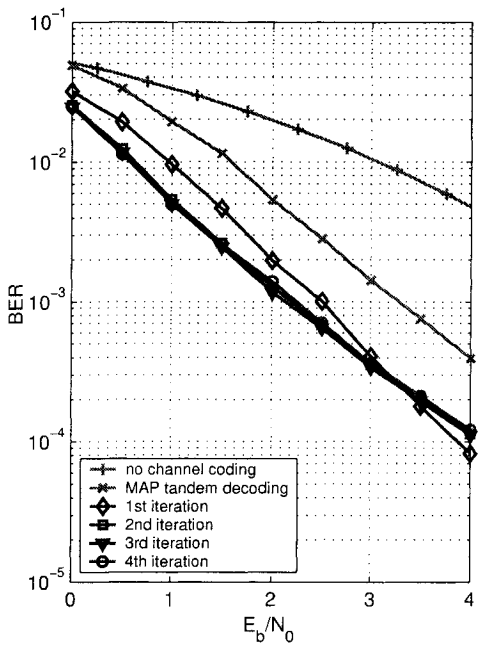
Here we considered the case where the joint decoder does not have access to the knowledge of the transmitted symbol sequence length $N = 50$. Figure 4.12 shows the obtained result. The first two graphs on top show the BER and SER, from left to right, of our proposed iterative scheme whilst the two bottom graphs show the BER and SER, from left to right, for the iterative scheme in [20].



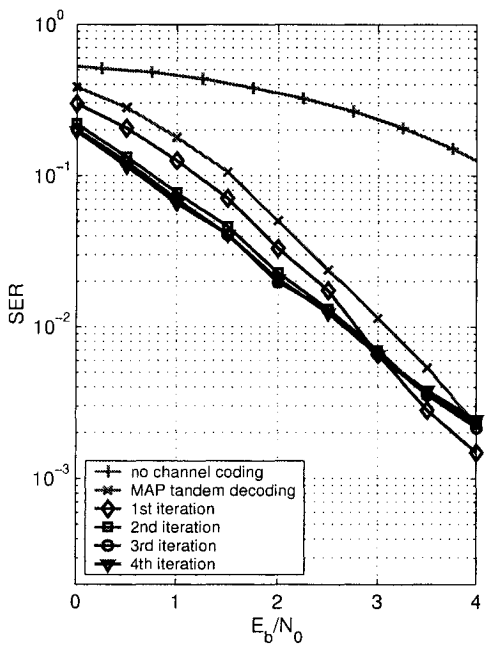
(a) BER for the proposed iterative scheme.



(b) SER for the proposed iterative scheme.



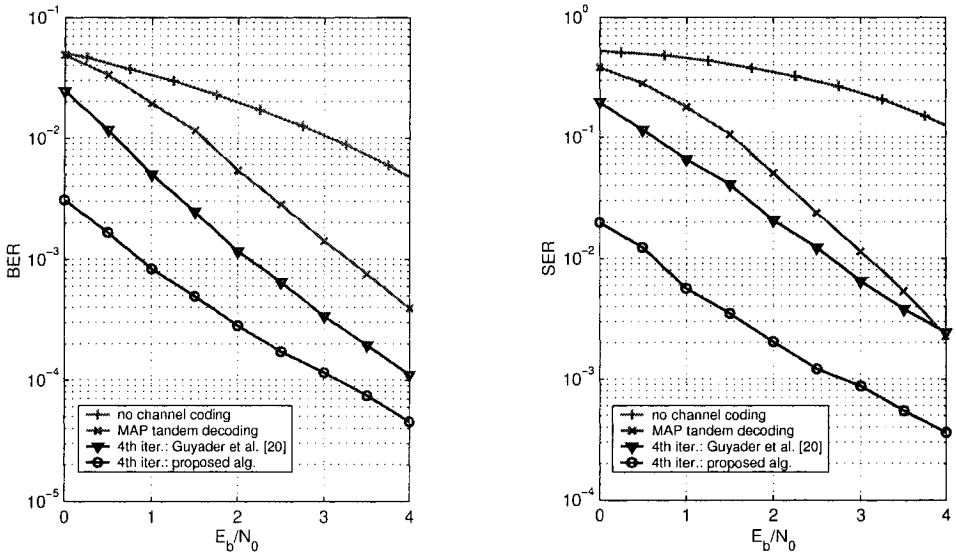
(c) BER for the iterative scheme of [20].



(d) SER for the iterative scheme of [20].

Fig. 4.12 BER and SER for different E_b/N_0 (coded) and for both the proposed iterative scheme and the one in [20]. The joint decoder has no knowledge of $N = 50$.

We can immediately note that the scheme proposed by Guyader et al.[20] suffers from a lack of convergence, with respect to the iteration. Indeed it can be seen that around 3.5 dB iterations 2 through 4 actually yield a worse performance than the first iteration. On the other hand, the proposed algorithm suffers from no such issue and exhibits either a gain or no gain at all from one iteration to the next. In call cases, it seems that little gains can be achieved by subsequent iterations (after the fourth that is). It is relatively easy to see that the proposed algorithm significantly outperforms that of [20] and has far greater synchronization power. Figure 4.13 compares the fourth iterations of both schemes in terms of BER and SER.



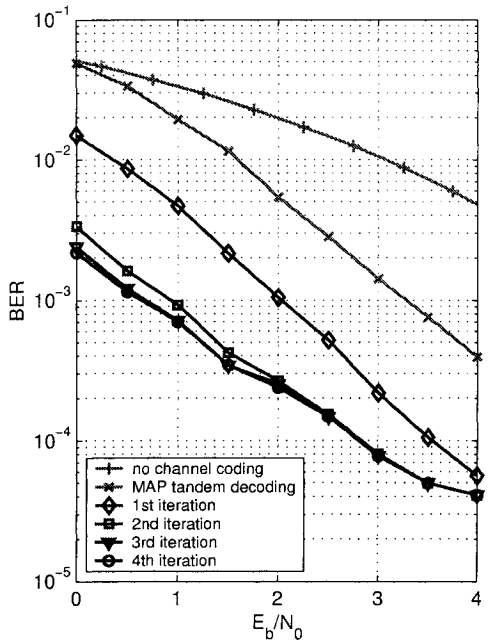
(a) BER vs.E_b/N_0 (coded). (b) SER vs.E_b/N_0 (coded).

Fig. 4.13 Comparing the fourth iteration of the proposed scheme with that in [20]. BER and SER for different E_b/N_0 (coded). The joint decoder has no knowledge of $N = 50$.

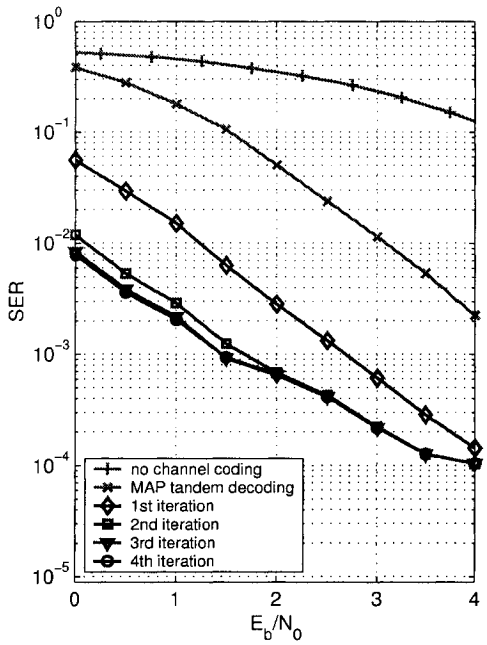
We note that, at the fourth iteration, our algorithm comes with a gain of 1.5 dB in BER and more importantly a gain of 2 dB in SER when compared with the fourth iteration of [20]. We recall that this gain is in fact obtained whilst at the same time *reducing* the number of performed computations. In the case of our simulations, where an efficient implemention of Belief Propagation was employed and an 8-symbols source used, the gain was obtained with a reduction of 45% in the number of multiplications and 30% in the number of additions.

Joint Decoder incorporates knowledge of N

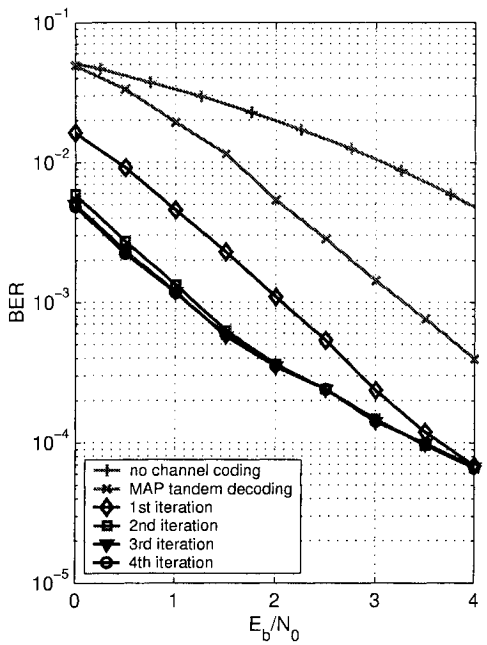
Here we considered the case where the joint decoder *has* access to the knowledge of the transmitted symbol sequence length $N = 50$. Figure 4.14 shows the obtained result. The first two graphs on



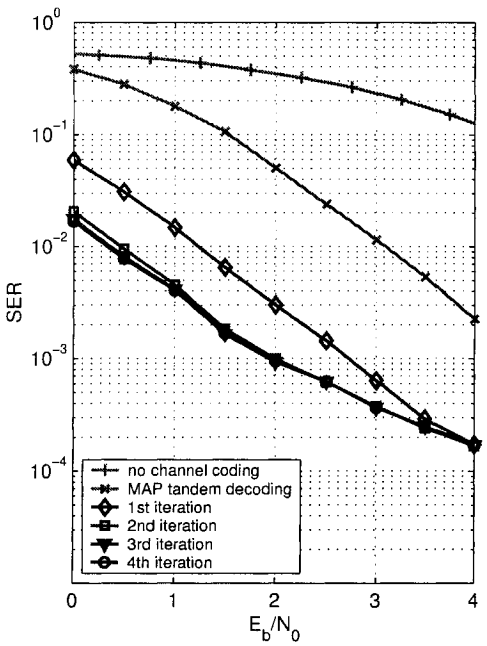
(a) BER for the proposed iterative scheme.



(b) SER for the proposed iterative scheme.



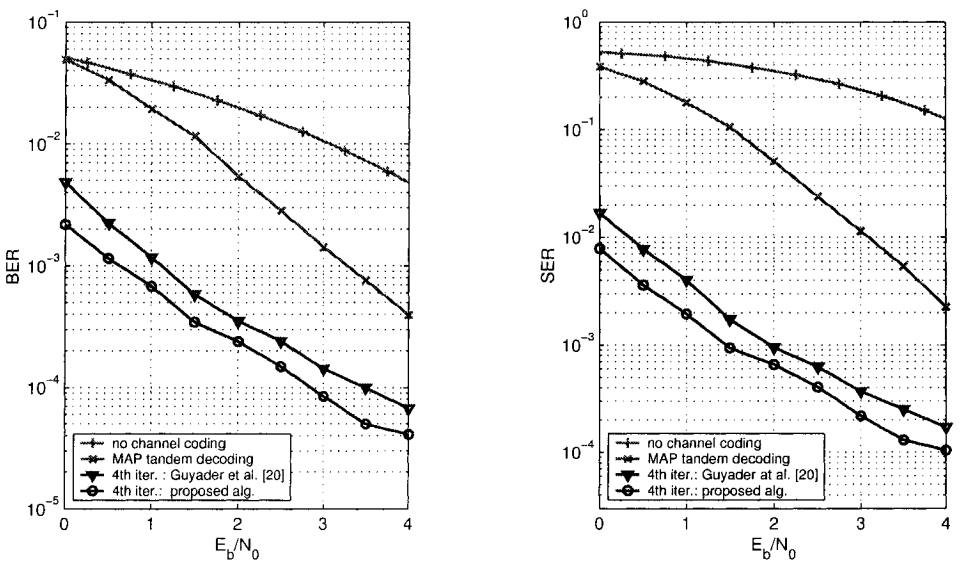
(c) BER for the iterative scheme of [20].



(d) BER for the iterative scheme of [20].

Fig. 4.14 BER and SER for different E_b/N_0 (coded) and for both the proposed iterative scheme and the one in [20] . The joint decoder has knowledge of $N = 50$.

top show the BER and SER, from left to right, of our proposed iterative scheme whilst the two graphs on bottom show the BER and SER, for the iterative scheme in [20]. Interestingly, it seems that incorporating knowledge of N has a significant effect on Guyader et al.'s [20] algorithm. Indeed, it does seem that this added constraint has rectified the earlier observed problem of convergence. Recalling the discussion in section 4.2.1, the algorithm in [20] imposed an assumption with respect to the independence of messages which is violated in actuality and yields to incorrect posterior probabilities for the bits U_k and variables X_k. It is likely that the added knowledge of N compensates for this problem yielding better approximation of the posterior probabilities in question. Again we can note that in all cases, very little gains are incurred after the second iteration. This being said, we can note that under the additional knowledge of $N = 50$, our algorithm still outperforms that in [20] as shown in figure 4.15. At the fourth iteration, we note

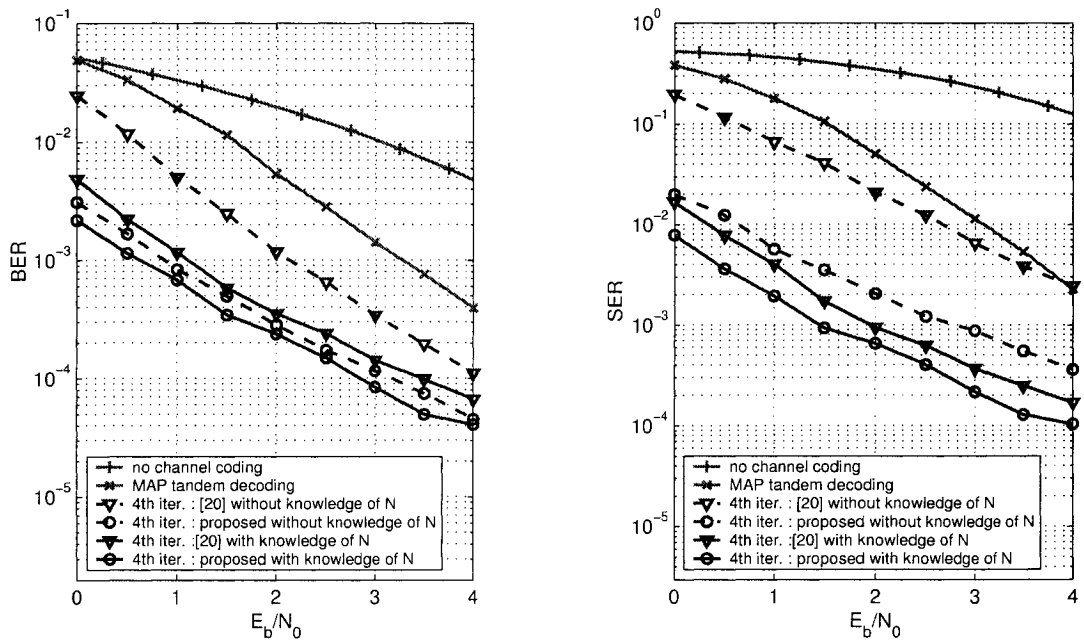


(a) BER vs.E_b/N_0 (coded). (b) SER vs.E_b/N_0 (coded).

Fig. 4.15 Comparing the fourth iteration of the proposed scheme with that in [20]. BER and SER for different E_b/N_0 (coded). The joint decoder has knowledge of $N = 50$.

a gain of approximately 0.5 dB for both BER and SER with respect to the fourth iteration of [20]. When using an efficient implementation of Belief Propagation this gain is obtained with a reduction of 53% in the multiplications and 39% in the addition operations performed by an efficient implementation of [20].

With respect with the previous results, the algorithm in [20] shows very significant gains (relatively to itself), whereas our proposed algorithm shows more modest gains. Figure 4.16 compares the fourth iterations of all cases for reference. Note the interesting phenomena that the BER for the



(a) BER vs.E_b/N_0 (coded). (b) SER vs.E_b/N_0 (coded).

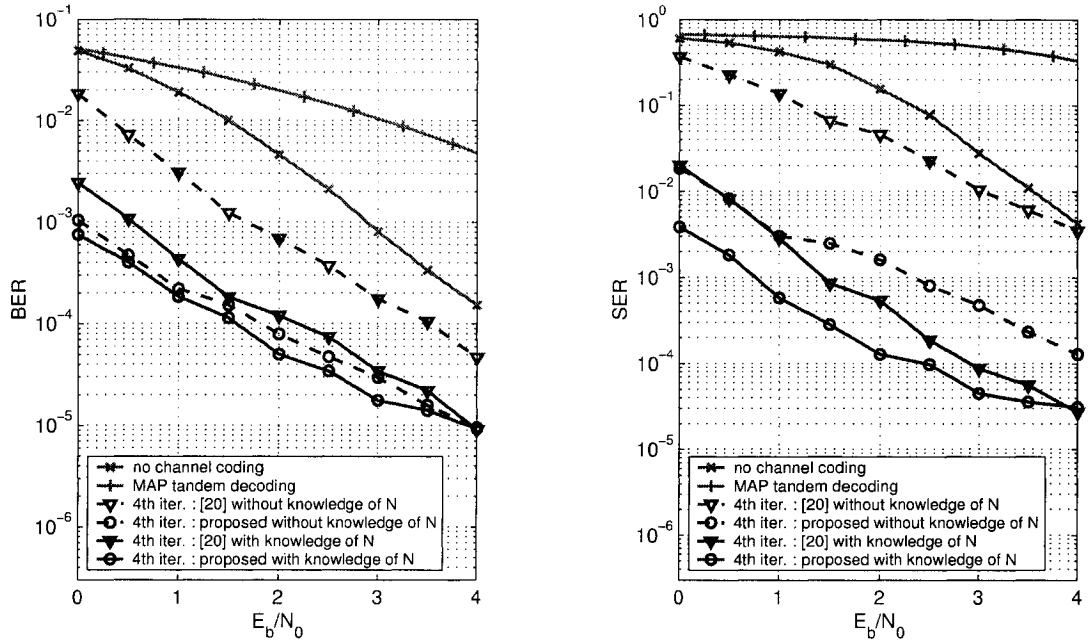
Fig. 4.16 Comparing the fourth iteration, all cases, of the proposed scheme and that in [20]. BER and SER for different E_b/N_0 (coded).Blocks of $N = 50$ symbols.

proposed without knowledge of N is slightly improved when compared with [20] that includes knowledge of N. Whereas the corresponding SER curves are reversed. This is attributed to the fact that the algorithm in [20] with knowledge of N has greater synchronization than our own with no knowledge of N. This phenomena also shows more generally that BER is not as appropriate a measure as SER when dealing with VLC's: indeed only one bit error may result in de-synchronization and lead to multiple symbol errors.

4.3.2 Blocks of 200 symbols

The same simulation were also carried out with blocks of $N = 200$ symbols decoded at a time, in order to evidence the gains obtained with larger decoding block length. The results are shown in Appendix B. In general all graphs show gains with respect to their $N = 50$ counterparts. This is expected and is the case for any decoding scheme, namely that decoding performance improves with larger blocks of data decoded at a time. The same general trends as for the corresponding

case for $N = 50$ may be observed. Figure 4.17 succinctly shows the fourth iterations of all cases.



(a) BER vs.E_b/N_0 (coded). (b) SER vs.E_b/N_0 (coded).

Fig. 4.17 Comparing the fourth iteration, all cases, of the proposed scheme and that in [20]. BER and SER for different E_b/N_0 (coded). Blocks of $N = 200$ symbols.

Looking at the case where the joint decoder has no knowledge of $N = 200$, we note that our algorithm provides with an additional gain of 1.5 dB for the BER and 3 dB for the SER when compared with [20]. Note that this gain was obtained while performing with a reduction of 45% in the number of multiplications and 30% in the number of additions (for efficient implementations).

Looking at the case where knowledge of N is incorporated, our algorithm provides with a gain of 0.5 dB in the BER and 1 dB in the SER. This with the asymptotic reduction of 53% in the multiplications and 39% in the addition operations (for efficient implementations).

Note also that the aforementioned phenomena — with respect to the reversal of the SER curves when comparing the proposed algorithm with no knowledge of N and the algorithm in [20] with knowledge of N — is still observed.

4.4 In Retrospect

The proposed algorithm has been shown to yield significantly better results along with a drastic reduction in computational complexity when compared to the existing one [20]. The synchronization power of our algorithm when the joint decoder does not incorporate knowledge of N is great enough to consider the possibility of using it in that case alone. Since incorporating knowledge of N, results in an N-fold increase in operations for the case that an efficient implementation is employed, this would represent a tremendous advantage in terms of complexity.

Consider figure 4.16(b) which shows the SER's for the fourth iterations of both algorithms and in all cases for $N = 50$. We note that when compared with the algorithm in [20] that includes knowledge of N, the proposed algorithm with no knowledge of N shows a performance loss of approximately 1 dB at high E_b/N_0 and less then 0.25 dB at low E_b/N_0. However, this minimal loss comes with the benefit that the proposed algorithm is performing as little as 1% of the operations when an efficient implementation is used in both cases. On the other hand, comparing with the proposed algorithm that includes knowledge of N, we note a performance loss of 0.75 dB at low E_b/N_0 and 1.5 dB at high E_b/N_0: in this scenario, the proposed algorithm with no knowledge of N is performing as little as 2% of the operations when an efficient implementation is used in both cases.

An even stronger case can be made by considering figure 4.17(b) for $N = 200$. If we opt to use the proposed algorithm with no knowledge of N, there is no loss at low E_b/N_0 and a loss of 1 dB at high E_b/N_0 when compared with [20] with knowledge of N. However in this scenario, the proposed algorithm is performing approximately as little as 0.25% of the operations when an efficient implementation is used in both cases..

Chapter 5

Other Improvements and Preliminary Results

In this chapter, we briefly present some preliminary ideas and results with respect to the proposed algorithm of the previous chapter. Many of the suggestions are not inter-related and it was therefore difficult to impose a structure upon them that would permit a natural flow of thought. For this reason, each of the following sections should be viewed as much as possible independently from one another. We begin with a brief section on the possibility of further reduction in computational complexity with no loss in performance, particulary in the case where the joint decoder has knowledge of N. Next, we show, more generally, how significant computational savings may be obtained with negligible losses in performance: this involves mainly, performing early hard decisions on those information bits deemed reliable. Next, we present preliminary results on the effects of the joint decoder possessing either no knowledge of the symbol transitions probabilities $P(s_n|s_{n-1})$ or incomplete knowledge. This section is followed by a brief discussion, along with preliminary results, on the effects of the recursive convolutional channel code and the interleaver. Finally, we show how an equivalent graph representing the coding chain may be built.

5.1 Further Reductions in Computational Complexity

We have already mentioned that when knowledge of the received symbol sequence length, N, is included in the joint decoder, we have $|\mathcal{X}| = N|\mathcal{S}||\mathcal{T}|$ which results in an N-fold increase in computations for the case that en efficient implementation of belief propagation is used. Indeed, the probability matrix $P(X_k|X_{k-1})$, shown in Appendix A, then contains $2N|\mathcal{S}||\mathcal{T}|$ and the matrix $P(U_k|X_k)$ contains $N|\mathcal{S}||\mathcal{T}|$ non-zero entries. Hence when node X_k activates, in an efficient implementation, it will perform its operations on all of those non-zero entries and hence assumes

that any state $X_k = (\gamma, v, c)$ is possible. The same applies for node U_k which will consider all possibilities of X_k. However it is clear for example that if the minimum codeword length of the source coder is $l_{min} = 10$, then node X_5 need not verify against states (γ, v, c) with $c \neq 0$. In general, given a minimum codeword length l_{min} and a maximum codeword length l_{max}, node X_k need only verify against states (γ, v, c) with,

$$\left\lceil \frac{k}{l_{max}} \right\rceil \leq c \leq \left\lceil \frac{k}{l_{min}} \right\rceil \tag{5.1}$$

with $k = 1, \ldots, K$. The same applies for node U_k. In essence, in terms of the matrix $P(X_k|X_{k-1})$ shown in appendix A, each node X_k should perform its operations on only those non-zero entries found in column $\lceil k/l_{max} \rceil$ to column $\lceil k/l_{min} \rceil + 1$. Again the same applies to U_k with its $P(U_k|X_k)$ matrix. In other words, X_k need only perform operations on $2(\lceil k/l_{min} \rceil - \lceil k/l_{max} \rceil + 1)|\mathcal{S}||\mathcal{T}|$ entries as opposed to $2N|\mathcal{S}||\mathcal{T}|$ and U_k need only perform operations on $(\lceil k/l_{min} \rceil - \lceil k/l_{max} \rceil + 1)|\mathcal{S}||\mathcal{T}|$ entries as opposed to $N|\mathcal{S}||\mathcal{T}|$ entries. This represents a tremendous reduction in computational complexity for the Markov source and source coder model. This would represent a decrease in operations, with respect to the efficient implementation, by a factor of

$$\frac{\lceil \frac{k}{l_{min}} \rceil - \lceil \frac{k}{l_{max}} \rceil + 1}{N} \tag{5.2}$$

per level k, that is for X_k and U_k. Note that this is not hard to implement and requires only that each link $X_{k-1} \rightarrow X_k$ be quantified not with $P(X_k|X_{k-1})$ but rather with the appropriate sub-matrix. And the same holds for the $X_k \rightarrow U_k$ link. The sub-matrices may all be determined once *offline* and come with the added benefit of *decreasing* the overall storage space. It is difficult to exactly determine overall reduction in complexity of Markov source and source coder. Note however, that the percentage computations equation above is more or less linearly increasing with k, hence a rough approximation of the overall average percentage computations,

$$\frac{\lceil \frac{N/2}{l_{min}} \rceil - \lceil \frac{N/2}{l_{max}} \rceil + 1}{N/2} \tag{5.3}$$

For the simulated 8-symbol source that exhibited $l_{min} = 2$, $l_{max} = 6$ and with $N = 200$ results in a decrease in computations by 65% from the efficient implementation at no cost in performance. Yet another possibility is to consider the possible values for c_k from the *end* of the data stream, which knowledge of N, here assumed, allows. In general, with this consideration we will have,

$$N - \left\lfloor \frac{K-k}{l_{min}} \right\rfloor \leq c_k \leq N - \left\lfloor \frac{K-k}{l_{max}} \right\rfloor \tag{5.4}$$

with $k = 1, \ldots, K$. Combining equations 5.4 and 5.1, we have,

$$\max\left\{\left\lceil\frac{k}{l_{max}}\right\rceil, N - \left\lfloor\frac{K-k}{l_{min}}\right\rfloor\right\} \leq c_k \leq \min\left\{\left\lceil\frac{k}{l_{min}}\right\rceil, N - \left\lfloor\frac{K-k}{l_{max}}\right\rfloor\right\} \tag{5.5}$$

with all of the previous developments holding. The only difference is that the possible values for c_k are then also a function of the received bit sequence length K, which in turn implies that we must compute a set of sub-matrices for each possible value of K. Although this may be done offline, it would unfortunately represent a high increase in storage space. On the other hand, the computational savings that such an implementation affords are in the neighborhood of 95%.

5.2 Bit Simplification

Another interesting idea for decreasing complexity, comes from the fact that some of the loops in the equivalent graphs may be *broken*[1] if an information bit were to be declared known and accordingly set, via a hard decision, to a specific value of 0 or 1. This is shown is figure 5.1. In

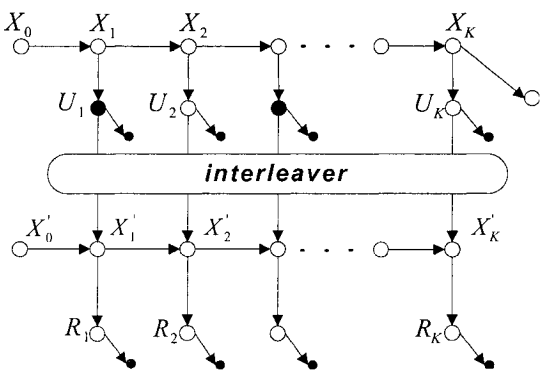


Fig. 5.1 Bit Simplification

effect those information bits that are so to speak simplified would now deliver constant messages to their neighbors and need not be activated any further. The question of when to simplify these bits arises. The most natural choices, given the observed performance of the proposed algorithm, is either before the first iteration or before the second iteration. Formally, we let $\theta \in [0, 1]$ represent a threshold. Simplifying the information bits before the first iteration entails that we perform,

$$\hat{u_k} = \arg\max_{u_k} P^0(u_k|y_k) \tag{5.6}$$

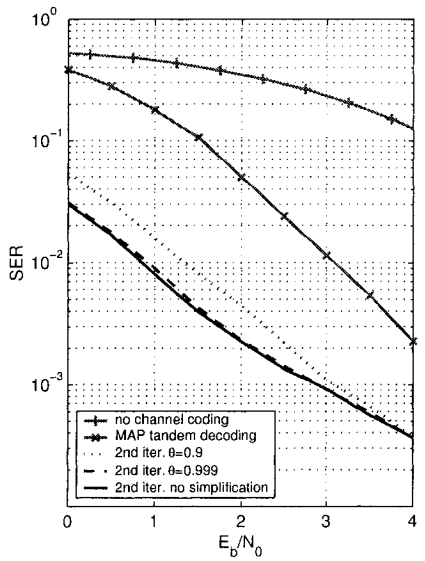
[1] recall the graphical separation criterion of section 2.2.2 which states that if the linking node is instantiated in a Head-to-Tail configuration $X_k \rightarrow U_k \rightarrow X'_k$, then $P(X_k|X'_k) = P(X_k)$.

$$P^0(u_k|y_k) = \begin{cases} \delta_{u_k,\hat{u_k}} & \text{if } \max_{u_k} P^0(u_k|y_k) > \theta \\ P^0(u_k|y_k) & \text{otherwise} \end{cases}$$

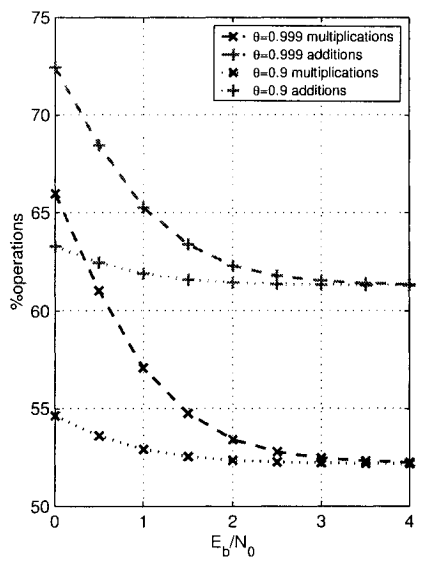
and the same hold with $P^1(u_k|y,z)$ if we were to simplify before the second iteration begins. Letting η represent the total number of bits simplified, such a scheme would result in a percentage operations (for an efficient implementation),

$$\frac{15K|\mathcal{X}| - 7\eta|\mathcal{X}| + 16K|\mathcal{X}'| - 8\eta|\mathcal{X}'|}{15K|\mathcal{X}| + 16K|\mathcal{X}'|}\ (\times) \qquad \frac{11K|\mathcal{X}| - 5\eta|\mathcal{X}| + 9K|\mathcal{X}'| - 2\eta|\mathcal{X}'|}{11K|\mathcal{X}| + 9K|\mathcal{X}'|}\ (+) \quad (5.7)$$

where $(\times)$ denotes multiplication and $(+)$ denotes addition operations. Clearly the performance will depend on the chosen threshold θ. Some preliminary results are shown in figure 5.2. We have



(a) Performance with thresholds $\theta = 0.9$ and $\theta = 0.999$.



(b) Percentage computations as a function of the E_b/N_0 points.

Fig. 5.2 Bit simplification before the second iteration with Thresholds $\theta = 0.9$ and $\theta = 0.999$ compared with no bit simplification.

considered the case of the previously simulated 8-symbol Gauss-Markov source, and the joint decoder was assumed not to have knowledge of $N = 50$. Further, we considered effecting the bit simplification before the second iteration. Interestingly, we have the surprising result that with $\theta = 0.999$, a negligible loss in performance (approximately 0.05 dB) is obtained with a significant reduction in computations. This behavior is expected the carry through more generally and may essentially be explained by the fact that the belief's of most information bits converge quickly and with high confidence to either the value of 1 or 0. However more simulations are needed in

order to substantiate such a method as well as to verify the behavior of the joint decoder if we were to effect the bit simplifications prior to the first iteration.

5.3 The Effects of Inexact Knowledge of $P(s_n|s_{n-1})$

Recall that our joint decoder, in order to have a fully specified Bayesian network representation of the coding chain, must have access to $P(s_n|s_{n-1})$. Such an assumption represents a severe constraint in practice since this information is generally not available at the receiver. An investigation with respect to the sensitivity of the proposed joint decoding algorithm to inexact knowledge of $P(s_n|s_{n-1})$ presents a natural extension. Figure 5.3 shows some preliminary results, again 8-symbol source in the case that the joint decoder has no knowledge of $N = 50$. When a memoryless source is assumed (modeling the case where the receiver has no knowledge of $P(s_n|s_{n-1})$ whatsoever), the decrease in performance is severe, underlining the importance of inter-symbol correlation in the performance of our algorithm. However, we also simulated the case where the $\hat{P}(s_n|s_{n-1})$ available at the receiver is given by,

$$\hat{P}(s_n|s_{n-1}) = \begin{cases} \alpha_n & P(s_n|s_{n-1}) > 0 \\ 0 & P(s_n|s_{n-1}) = 0 \end{cases}$$

where α_n is such that $\sum_{s_n} \alpha_n = 1$. In words, the receiver is aware of which symbols s_n are *impossible* given s_{n-1} and those symbols that are possible are simply assumed to be equally likely. Therefore, the only *true* information that the receiver has access to is the impossible symbol transitions. In

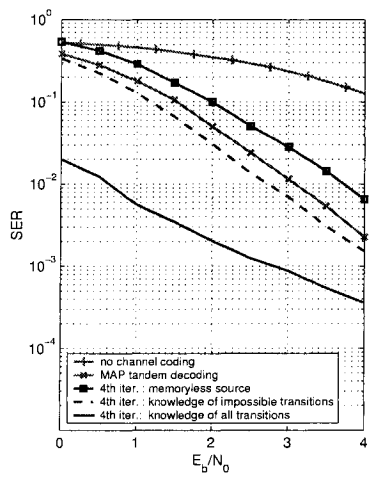


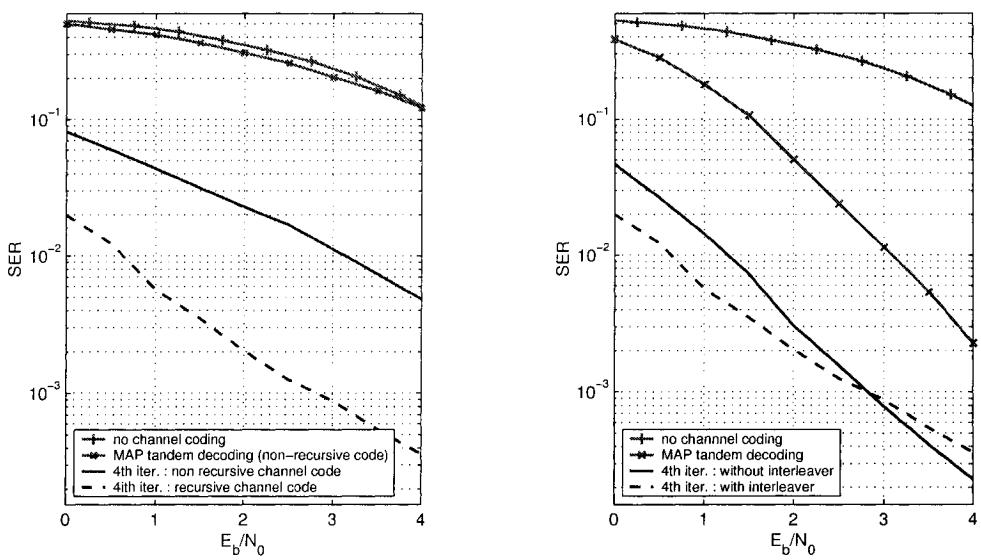
Fig. 5.3 The effects of inexact Knowledge of $P(s_n|s_{n-1})$.

this case, the performance decrease is less severe. It is the belief of this author that adapting Belief

Propagation so that it maximizes the posterior probability on the *sequence* of information bits (as in Viterbi Algorithm) would result in an improved performance for these simulated cases of a source code of the VLC variety. Indeed, the algorithm will then likely have more synchronization power. More analysis is needed with respect to this idea.

5.4 The Effects of the Interleaver and Recursive Convolutional Code

Studying the effects of the interleaver and the recursive convolutional code may very well lead to significant insight with respect to turbo-decoding in general. Recall that we have originally defined joint decoding as a problem that may treated separately from the problem of joint coding. The idea was that joint decoding may be performed on systems that employ tandem decoding in order to take into account the residual redundancy of the source coder and the source memory. However, in our search for such a joint decoder from the Bayesian network setting, it became apparent that the insertion of an interleaver between the source coder and the channel coder would result in a better performance. If the combination of an interleaver and recursive channel coder is necessary in order to obtain gains with respect to the tandem decoding strategy, this would imply that we have actually designed a joint source channel coding/decoding scheme rather than a joint decoder. Preliminary results in figure 5.4, where the 8-symbol source is simulated for the case that the joint decoder has no knowledge of $N = 50$, indicate otherwise. Indeed, the use of



(a) Placing a non-recursive systematic convolutional channel coder.

(b) Removing the interleaver.

Fig. 5.4 The effects of the interleaver and recursive convolutional code.

a non-recursive convolutional still yields significant gain with respect to the corresponding MAP tandem decoding. Removing the interleaver has an interesting effect: with a loss at low E_b/N_0 and a small gain at high E_b/N_0. This most likely indicates a poor design choice for our interleaver. Interesting work lies in the effect of an improved interleaver design.

5.5 Anti-Causal Graph

Finally, note that the Bayesian network representation of the coding chain was in all cases derived assuming the natural ordering of the random variables imposed by causal time. An interesting idea is then to consider other possible orderings with their resulting graphs. One such ordering is to assume (note that this is extremely counter-intuitive) the channel coding operation to happen before the source coding operation. The ordering on the random variables is then X',R,U,X. The resulting graph is shown in figure 5.5. Interestingly this graph has the same advantageous

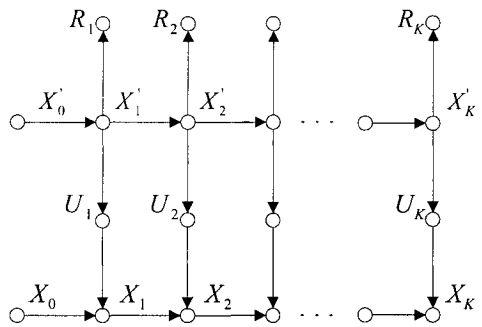


Fig. 5.5 Anti-causal graph.

properties as the proposed one: it holds the same state-space, exhibits the same number of loops and is as well reduced in computational complexity when used for decoding. It is expected that the same gains will be achieved.

Chapter 6

Conclusion

We began this text by defining the problem of joint source-channel decoding separately from the joint coding problem. This was based on the simple premise that any *practical* system that utilizes tandem encoding must necessarily use finite block lengths for the source coder and the channel coder. This in turn implies that the received data stream possesses additional redundancies, namely the source memory and the residual redundancy of the source coder, that are ignored by a tandem decoding strategy. A natural consideration therefore is to consider the design of a joint decoder, specifically for such systems, that would take these additional sources of redundancies into account. In this context, the optimal joint decoding solution was exposed. Unfortunately such a solution, suffering from a state-space *explosion*, remains intractable, leading to the need for less complex and therefore sub-optimal joint decoders.

Second, we defined Bayesian networks and saw that they essentially provide an intuitive graphical framework for the analysis of statistical problems. The algorithm of Belief Propagation, derived from first principles, was seen to represent an efficient solution to the inference problem; a solution that is guaranteed to converge in linear time as long as it is applied to singly connected graphs. Moreover, we saw that Belief Propagation is a generalization of the forward-backward algorithm and can be used to provide maximum a-posteriori estimates in the context of decoding. Finally, we mentioned the still misunderstood phenomenon of applying Belief Propagation to non-singly connected graphs and obtaining approximate posterior probabilities.

Next, we saw how the joint decoding problem may be approached and analyzed from the context of Bayesian networks as previously done in [20]. Specifically, the Bayesian network representation of the coding chain — namely the source, the source coder and the channel coder — was first derived. Subsequently, we saw that the resulting graph along with the corresponding conditional probability measures may be made available to the receiver as long as the latter has access to the source statistics. With a fully specified graph of the coding chain available, the

receiver may then implement a joint decoding scheme, by incorporating the received data and possibly the length of the received symbol sequence, and applying the algorithm of Belief Propagation. Unfortunately it was noted that the graph is non-singly connected and in fact contains a significant number of undirected cycles. The iterative solution proposed by [20] was then exposed. This entailed inserting an interleaver and splitting the graphs into two sub-graphs — one for the Markov source and source coder model and another for the channel coder model. The iterative scheme consisted of performing Belief Propagation on the channel coder model, passing externally computed extrinsic information quantities, performing Belief Propagation on the Markov source and source coder model and finally passing extrinsic information quantities back to the channel coder model and so on. In terms of the larger context of joint decoding, this solution was seen to be a sub-optimal, limited complexity, joint decoding scheme taking into account three types of redundancies present in the coding chain: the source memory, the source coder's residual redundancy as well the redundancy introduced by the channel coder.

Finally, we presented a new joint source-channel decoder that is largely inspired from the previously outlined developments. The algorithm's novelty was first based on deriving an equivalent Bayesian network representation of the coding chain. This new representation relied on the simple yet potent observation that the information bits depend on the given state of the Markov source and source coder model and not on the transitions of states. Second, we derived a new methodology for effecting the iterations. Specifically, we showed that the iterative scheme may be implemented simply as a specific ordering of node activation. This has the advantage of forgoing the additional overhead of separating the graph and computing the extrinsic information quantities externally to the Belief Propagation process. The theoretical analysis that followed showed that the proposed equivalent graph possesses improved convergence properties when compared with [20] as it not only relaxes a stringent assumption but it also contains a mere fraction of the loops. It was also seen that the proposed algorithm has a significantly reduced computational complexity. Finally, computer simulations substantiated our analysis as gains of up to several decibels in the symbol error rate were observed along with a drastic reduction in computational complexity when compared with [20].

Additional possibilities for improvement were also outlined and some preliminary results presented. Most interesting was the possibility of further reductions in computational complexities for the specific case where the joint decoder incorporates knowledge of the received symbol sequence length. It was seen that in such a case the state-variable of the Markov source and source coder model need only verify its probabilities against a subset of states as opposed to all states. This would represent a tremendous decrease in complexity at no cost in performance. Another interesting idea for decreasing complexity, this time with a negligible cost to performance, came from performing early hard decisions on those information bits deemed reliable with respect to a

simple criterion. Next, we saw that adapting Belief Propagation so that it maximizes the probability on the sequence of information bits, may lead to a better performance in the VLC case. We also saw that a better design of the interleaver will most likely result in a superior performance of the proposed algorithm in general, particularly at high E_b/N_0.

Seen in a larger context, the proposed algorithm may be used on any system that employs tandem encoding and would provide very substantial gains in performance with respect to the commonly used tandem decoding strategy. Although this improvement in performance does come at the cost of an increased *decoding* complexity, the proposed algorithm remains tractable. Further, as suggested in Chapter 5, there is still much room for improvements in terms of reducing the computational complexity of the proposed algorithm. With these suggestions implemented, the proposed algorithm presents a viable alternative to tandem decoding.

On a final note, since the proposed algorithm requires knowledge of the source statistics, a convenient application may be found in systems designed for the transmission of natural data such as speech and images. Indeed, these types of data posses a great deal of correlation and they are therefore amenable to probabilistic modeling. With such models available, the proposed algorithm is immediately implementable.

Appendix A

Computational Complexity Analysis

This appendix is intended to support the computational complexity analyses found in the text. In the first section we consider the number of multiplication and addition operations required by a single node activation with respect to the Belief Propagation algorithm. In the second section, we consider the number of operations required assuming the transition probability matrix of that node is sparse. Finally, we show the sparse nature of the matrices involved in the Bayesian network representation of the coding chain.

A.1 Black Box Implementation of Belief Propagation

Let node X represents a random variable taking values on the set $\mathcal{X}$. Further, let node X be connected to n parent nodes $U_1, \ldots, U_n$ and m child nodes $Y_1, \ldots, Y_m$. Let the n parents of X represent random variables taking values on the set $\mathcal{U}$. Finally, let the m children of X represent random variables taking values on the set $\mathcal{Y}$. The steps describing a node activation are found in section 2.3.1. Here, we compute the number of operations required for such a node activation. Certain assumption were made that violate in principle the supposition of a black box implementation. First, we assume that normalization of all quantities must be carried out. This is mainly because it was noted that without normalization Belief Propagation can quickly result in instability. Note also that the quantities $\pi(x)$ and $\lambda(x)$ computed in the first step are naturally not to be recomputed when needed later.

STEP 1—Belief updating

Node X must compute its belief according to,

$$\begin{aligned} BEL(x) &= \alpha\lambda(x)\pi(x) \\ &= \alpha\Big[\prod_{j=1}^{m}\lambda_{Y_j}(x)\Big]\Big[\sum_{\mathbf{u}}P(x|\mathbf{u})\prod_{i=1}^{n}\pi_X(u_i)\Big] \end{aligned} \tag{A.1}$$

with α such that $\sum_x BEL(x) = 1$. We note that the present form of the equation is an efficient factorization and as such we assume the equation to be implemented as written. The number of multiplication required is given by,

$$n|\mathcal{U}|^n|\mathcal{X}| + (m-1)|\mathcal{X}| + 2|\mathcal{X}| \tag{A.2}$$

where the first term corresponds the multiplications required for the computation of $\pi(x)$ or the second bracketed term in A.1. The second term corresponds to the necessary multiplications to compute $\lambda(x)$ or the first bracketed term in A.1, while the last term corresponds to the multiplications necessary for normalization. Now, the number of required additions is,

$$(|\mathcal{U}|^n - 1)|\mathcal{X}| + (|\mathcal{X}| - 1) \tag{A.3}$$

where the first term corresponds to the additions required in the computation of $\pi(x)$ and the last corresponds for the additions necessary for normalization.

STEP 2—Bottom-up propagation

Node X must compute n messages $\lambda_X(u_i)$ to be sent to each parent U_i according to

$$\lambda_X(u_i) = \beta\sum_x \lambda(x)\sum_{u_k:k\neq i} P(x|\mathbf{u})\prod_{k\neq i}\pi_X(u_k) \tag{A.4}$$

where β is such that $\sum_{u_i}\lambda_X(u_i) = 1$. Again the equation's form represents an efficient factorization. Here we assume the equation is to be implemented as written and we assume further that $\lambda(x)$ is already available since it was computed in the first step. Hence the number of required multiplications is given by,

$$n(n-1)|\mathcal{U}|^n|\mathcal{X}| + n|\mathcal{X}||\mathcal{U}| + n|\mathcal{U}| \tag{A.5}$$

where the first term corresponds to the number of multiplications required in the second summation term of A.4: this taking into account that those multiplications must be carried out for every value of x, every value of u and further for the n parents. The second term represents

the multiplications necessary for the first summation term of A.4 taking into account that they must be performed for every value of u and the n parents. Finally the last term represents the multiplications necessary to normalize all n messages. Similarly, the number of additions is found to be,

$$n(|\mathcal{U}|^{n-1} - 1)|\mathcal{X}||\mathcal{U}| + n(|\mathcal{X}| - 1)|\mathcal{U}| + n(|\mathcal{U}| - 1) \tag{A.6}$$

where each term corresponds back to A.4 as discussed.

STEP 3—Top-down propagation

Node X must finally compute m messages $\pi_{Y_j}(x)$ to be sent to each child Y_j

$$\pi_{Y_j}(x) = \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x) \tag{A.7}$$

with α such that $\sum \pi_{Y_j}(x)_x = 1$. Here we assume that $\pi(x)$ is already available since it was computed in the first step. Assuming m of these messages need to be computed, the number of multiplications is then simply given by,

$$m^2|\mathcal{X}| \tag{A.8}$$

And the number of additions is given by,

$$m(|\mathcal{X}| - 1) \tag{A.9}$$

Total Computations

The total computations, for the activation of node X is given by summing the appropriate equations and yields,

$$\begin{cases} n^2|\mathcal{U}|^n|\mathcal{X}| + n|\mathcal{U}|\big(|\mathcal{X}| + 1\big) + |\mathcal{X}|(m+1) + m^2|\mathcal{X}| & \text{multiplications} \\ (n+1)|\mathcal{U}|^n|\mathcal{X}| + m|\mathcal{X}| - (m+n+1) & \text{additions} \end{cases} \tag{A.10}$$

A.2 Efficient Implementation of Belief Propgation for Sparse Matrices

It is often the case that the conditional probability matrix quantifying a a node's links to its parent is sparse. In such a scenario, the most efficient implementation of Belief Propagation would simply consider summing those elements for which $P(x|u_1, \ldots, u_n)$ are non-zero. In this section, we consider the number of computations that would be required assuming that the conditional probability matrix has a total of ζ *non-zero* entries. We assume further that those non-zero entries are symmetrically distributed in the sense that any one-dimensional row or column of $P(x|u_1, \ldots, u_n)$ contains the same number of non-zero elements. We assume the operations necessary for normalization are to be included. Note again that the $\pi(x)$ and $\lambda(x)$ quantities are computed only once, say for the first step. The number of operations required for each step is given below.

STEP 1—Belief updating

$$\begin{aligned} BEL(x) &= \alpha\lambda(x)\pi(x) \\ &= \alpha\left[\prod_{j=1}^{m}\lambda_{Y_j}(x)\right]\left[\sum_{\mathbf{u}}P(x|\mathbf{u})\prod_{i=1}^{n}\pi_X(u_i)\right] \end{aligned} \tag{A.11}$$

The number of multiplications needed is,

$$n\zeta + (m-1)|\mathcal{X}| + 2|\mathcal{X}| \tag{A.12}$$

where the first term corresponds to the multiplications required in $\pi(x)$, the second term corresponds to the required multiplications of $\lambda(x)$ and the last term corresponds to the normalization operation. The number of additions required is given by,

$$(\zeta - 1) + (|\mathcal{X}| - 1) \tag{A.13}$$

where the first term again corresponds to $\pi(x)$ and the second term corresponds to the additions required in the normalization operation.

STEP 2—Bottom-up propagation

$$\lambda_X(u_i) = \beta\sum_{x}\lambda(x)\sum_{u_k:k\neq i}P(x|\mathbf{u})\prod_{k\neq i}\pi_X(u_k) \tag{A.14}$$

We recall that there are n such messages to be sent, one for each parent U_i. The number of required multiplications, recalling our symmetry assumption, is given by,

$$n^2\zeta + n|\mathcal{U}| \tag{A.15}$$

where the first term corresponds to the two summations and the last term corresponds to normalization. The required additions on the other hand are given by,

$$n\zeta + n(|\mathcal{U}| - 1) \tag{A.16}$$

and the terms correspond as before.

STEP 3—Top-down propagation

$$\pi_{Y_j}(x) \quad = \quad \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x) \tag{A.17}$$

Clearly, here the operations required are the same as in the black box case. Namely

$$m^2|\mathcal{X}| \tag{A.18}$$

multiplications and,

$$m(|\mathcal{X}| - 1) \tag{A.19}$$

additions.

Total Computations

The total computations, for the activation of node X assuming ζ non-zero entries for the conditional probability matrix are then,

$$\begin{cases} n(n+1)\zeta + n|\mathcal{U}| + (m^2+m+1)|\mathcal{X}| & \text{multiplications} \\ \\ (n+1)\zeta + n|\mathcal{U}| + (m+1)|\mathcal{X}| - (m+n+1) & \text{additions} \end{cases} \tag{A.20}$$

It is important note that such an implementation would require a storage space for 2ζ entries, that is one entry for the actual value of the non-zero entry in $P(x|u_1,\ldots,u_n)$ and one entry to indicate the location of that non-zero entry in the matrix.

A.3 The Sparse Matrices in the Bayesian network representation of the coding chain

In this section, we examine the sparse nature of the matrices involved in the Bayesian network representation of the coding chain. We draw the reader's attention to the fact that, here, we break from the notation assumed in the previous sections and utilize instead the notation introduced in chapter 3. We begin by treating the case where the joint decoder does not assume knowledge of the received symbol sequence sequence length N, and study the corresponding matrices . Next, we show the case where knowledge of N is included. Both the proposed algorithm and the existing one [20] are treated.

A.3.1 Without knowledge of N

This case is relatively straight forward. Recall that X represents the state-space variable of the Markov source and source coder mode. Recall further that it was determined that $|\mathcal{X}| = |\mathcal{S}||\mathcal{T}|$, where $\mathcal{S}$ is the state-space of the source symbols and $\mathcal{T}$ is the set of all inner-vertices of the binary tree τ representing the source coder mapping. Finally the states of X are given by $(\gamma^{(i)}, v^{(j)})$, where $(\gamma^{(i)}$ represents the last completed symbol and $v^{(j)}$ represents the current vertex of *tau* with the exception that when $v^{(j)}$ is a leaf vertex, it is replaced by the root vertex $v^{(0)}$.

It should be immediately clear that each state of X is allowed at most two possible transitions, one for the next vertex upwards — representing an information bit of 1 —- and one for the next vertex downwards — representing an information bit of 0. It is indeed possible that the topology of τ allows less then 2 transitions for a given state or that the conditional probability matrix $P(s_n|s_{n-1})$ has zero entries in which case a transition out of a state may carry zero probability. However we assume here that in fact 2 transitions may occur for every state as this will provide us with an *upper bound* on the number of possible non-zero entries of the matrices. Hence, we have the immediate result that $P(X_k|X_{k-1})$ contains $2|\mathcal{X}|$ non-zero entries:

$$\zeta_{P(X_k|X_{k-1})} = 2|\mathcal{X}| = 2|\mathcal{S}||\mathcal{T}| \tag{A.21}$$

As for the probability matrix, $P(U_k|X_{k-1}, X_k)$, that is to quantify node U_k's link to its parents, X_{k-1} and X_k, in the scheme of Guyader et al. [20], we note from the previous discussion that there are only $2|\mathcal{X}|$ possible pairs. Each one of those pairs deterministically yields an information bit of either 1 or 0 and hence only one non-zero entry, per pair, is found in the matrix. This gives us therefore,

$$\zeta_{P(U_k|X_{k-1},X_k)} = 2|\mathcal{X}| = 2|\mathcal{S}||\mathcal{T}| \tag{A.22}$$

As for the probability matrix $P(U_k|X_k)$ that is to quantify the $X_k \rightarrow U_k$ link of our proposed scheme, we note that each of the states of X is deterministically associated with an information bit, hence we have,

$$\zeta_{P(U_k|X_k)} = |\mathcal{X}| = |\mathcal{S}||\mathcal{T}| \tag{A.23}$$

Finally, the channel coder is assumed to possess a sate-space $\mathcal{X}'$. Once again, we have the interesting result that since each state of the channel coder is allowed two transitions out depending on the current information *input* bit. Hence,

$$\zeta_{P(X'_{k+1}|U_k,X'_k)} = 2|\mathcal{X}'| \tag{A.24}$$

A.3.2 With knowledge of N

Recall that in this case, the state variable X is to be replaced with the new state variable $W = (X, C)$ where C is a variable representing the total *number* of completed symbols. However, the topology of τ imposes yet again the same constraint, namely that each state is allowed at most two possible transitions. Hence the previously determined upper-bounds all hold and we have now,

$$\zeta_{P(W_k|W_{k-1})} = 2|\mathcal{W}| \tag{A.25}$$

$$\zeta_{P(U_k|W_{k-1},W_k)} = 2|\mathcal{W}| \tag{A.26}$$

$$\zeta_{P(U_k|W_k)} = |\mathcal{W}| \tag{A.27}$$

$$\zeta_{P(X'_{k+1}|U_k,X'_k)} = 2|\mathcal{X}'| \tag{A.28}$$

In order to somehow clarify the situation,figure A.1 shows the relationship between $P(X_k|X_{k-1})$ and its corresponding $P(W_k|W_{k-1})$. The latter is formed by placing the matrix $P(X_k|X_{k-1})$ along the diagonal of $P(W_k|W_{k-1})$ and moving those non-zero entries that correspond to a symbol termination to the same state X but with an incremented counter C.

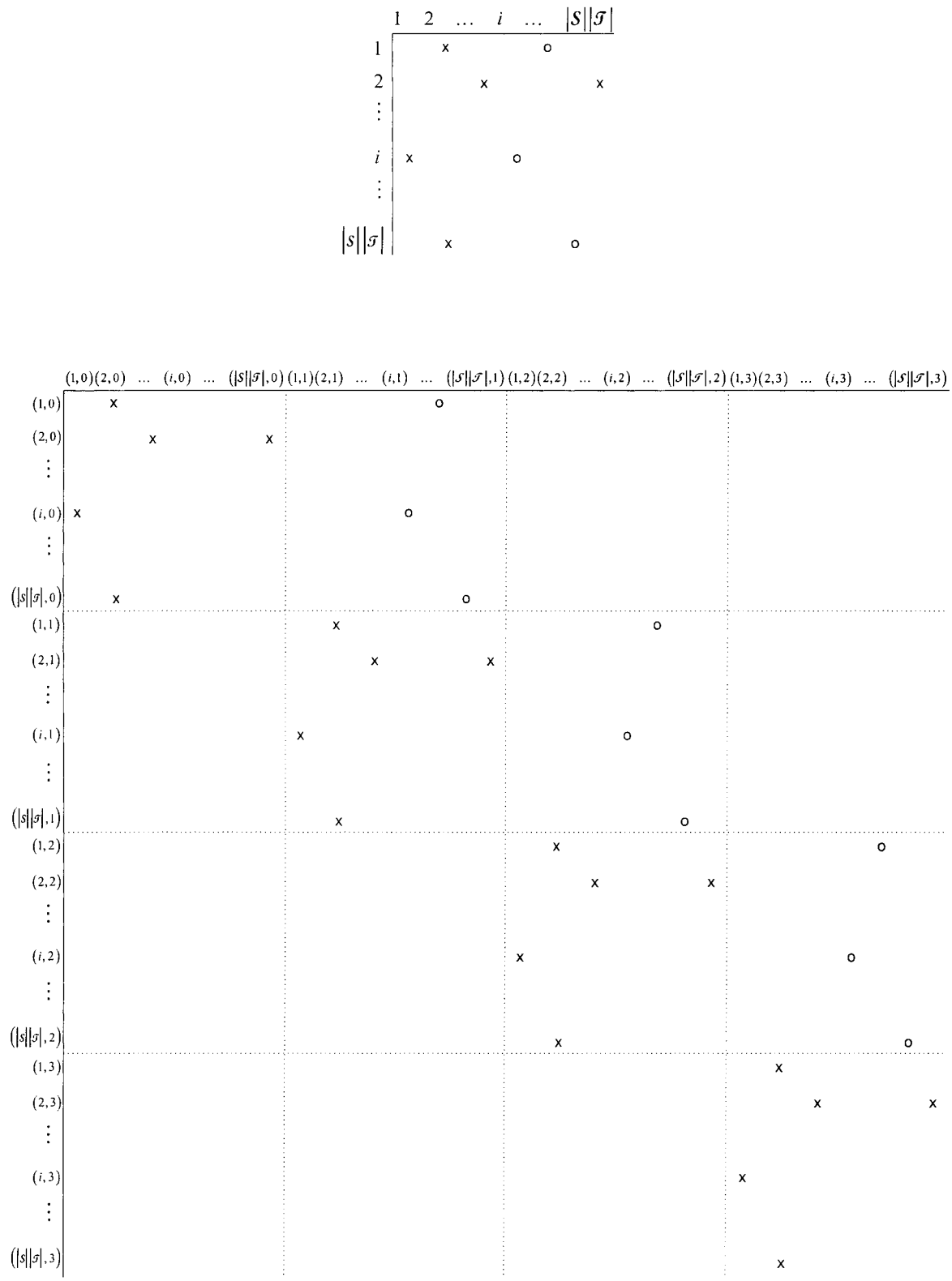


Fig. A.1 Realtionship between $P(X_k|X_{k-1})$ and its corresponding $P(W_k|W_{k-1})$. On top we show $P(X_k|X_{k-1})$ with two allowable transitions per state and hence two non-zero entries per row, some of which, indicated by 'o' correspond to a symbol completion. On the bottom, we show how to obtain $P(W_k|W_{k-1})$ from $P(X_k|X_{k-1})$. All empty spaces are zero entries

Hence, using our simplified notation with W denoted by X and the distinction with respect to the two is clarified by context, we have that the upper bounds to the non-zero entries of all matrices involved are given by,

$$\zeta_{P(X_k|X_{k-1})} = 2|\mathcal{X}| \tag{A.29}$$

$$\zeta_{P(U_k|X_{k-1},X_k)} = 2|\mathcal{X}| \tag{A.30}$$

$$\zeta_{P(U_k|X_k)} = |\mathcal{X}| \tag{A.31}$$

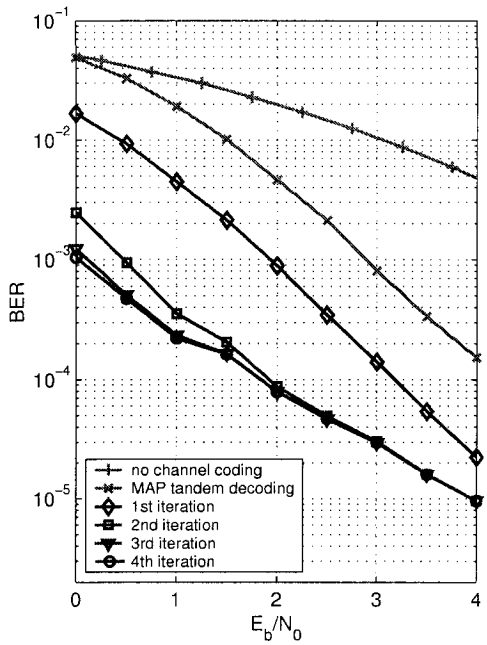
$$\zeta_{P(X'_{k+1}|U_k,X'_k)} = 2|\mathcal{X}'| \tag{A.32}$$

where $\mathcal{X} = |\mathcal{S}||\mathcal{T}|$ when no knowledge of N is incorporated and $\mathcal{X} = N|\mathcal{S}||\mathcal{T}|$ when knowledge of N is incorporated. And where $|\mathcal{X}'|$ is the state-space of the channel coder. On a final note, we point out that the non-zero entries in the matrices used in the Bayesian network representation of the coding chain are symmetrically distributed so that equation A.20 holds.

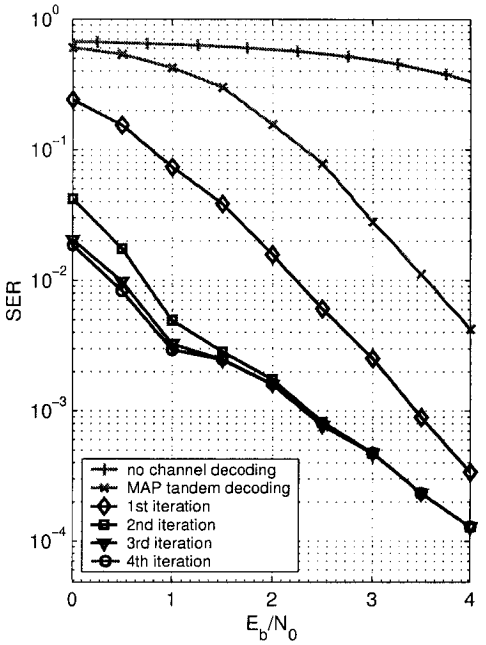
Appendix B

Additional Results

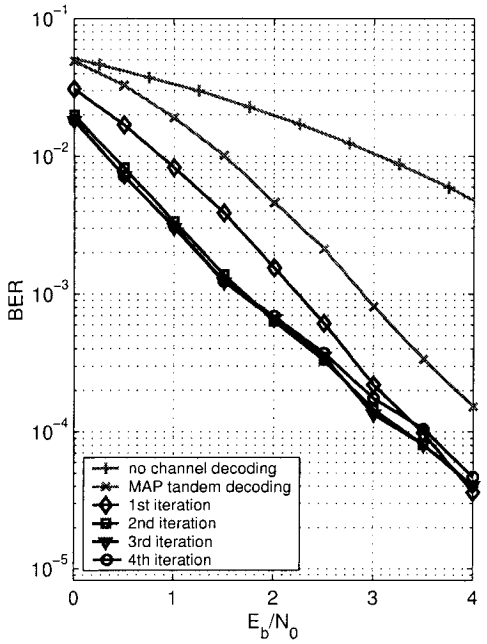
In this appendix, we present the complete results of 4.3.2 for the case that $N = 200$ blocks of data are decoded at a time. In all of the figures to follow, we plotted bit error rate (BER) and symbol error rate (SER) for different E_b/N_0 with E_b representing the *coded* bit energy. The first curve in all of the figures corresponds to the case where no channel coding is employed: the received bit stream is therefore hard decoded assuming independent bits, to obtain the BER, followed by a hard Huffman decoding to obtain the SER. The second curve represents the commonly used tandem decoding, namely MAP channel decoding assuming an input of independent bits, followed by hard Huffman decoding. The subsequent curves show the first to fourth iterations of either the proposed iterative scheme or that in [20] which was implemented verbatim.



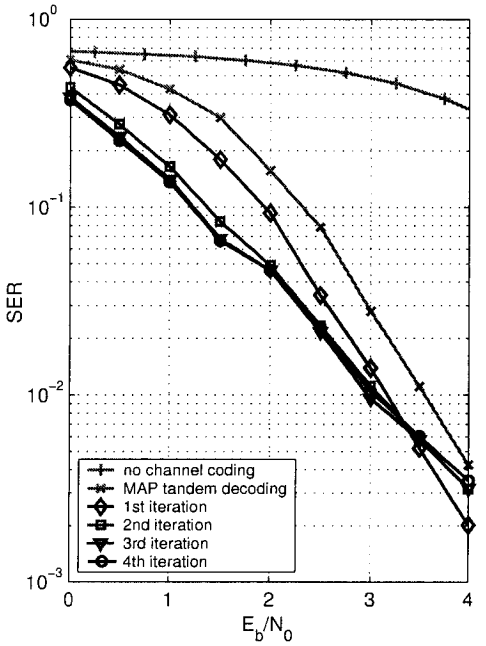
(a) BER for the proposed iterative scheme.



(b) SER for the proposed iterative scheme.

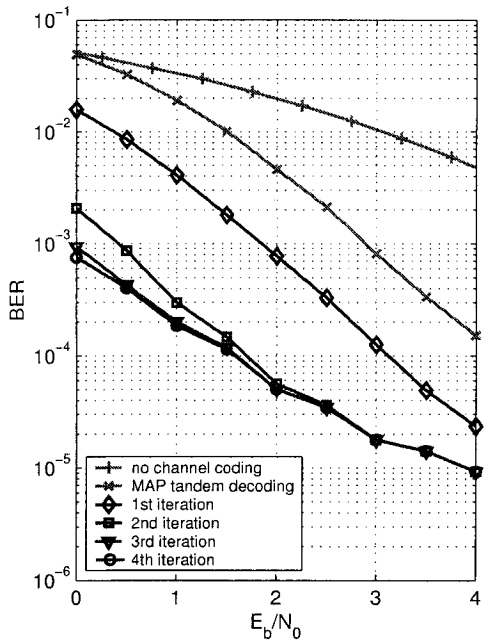


(c) BER for the iterative scheme of [20].

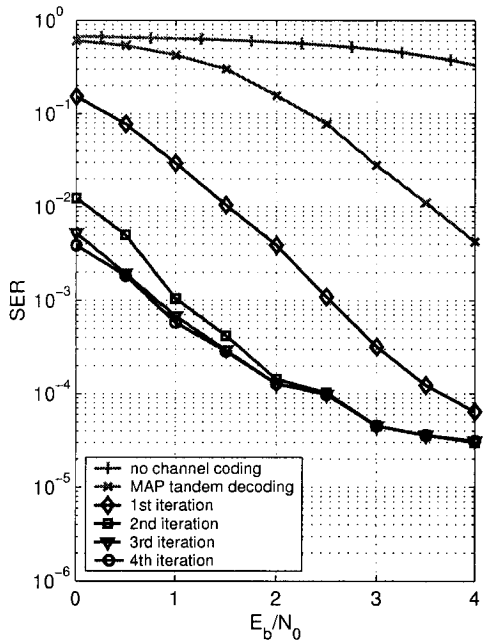


(d) SER for the iterative scheme of [20].

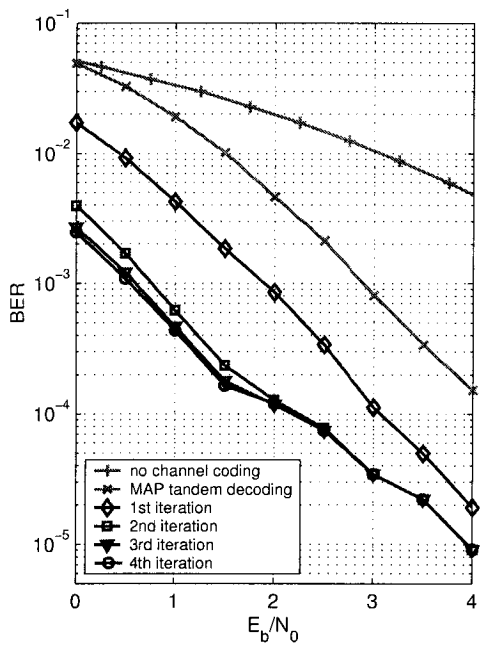
Fig. B.1 BER and SER for different E_b/N_0 (coded) and for both the proposed iterative scheme and the one in [20] . The joint decoder has no knowledge of $N = 200$.



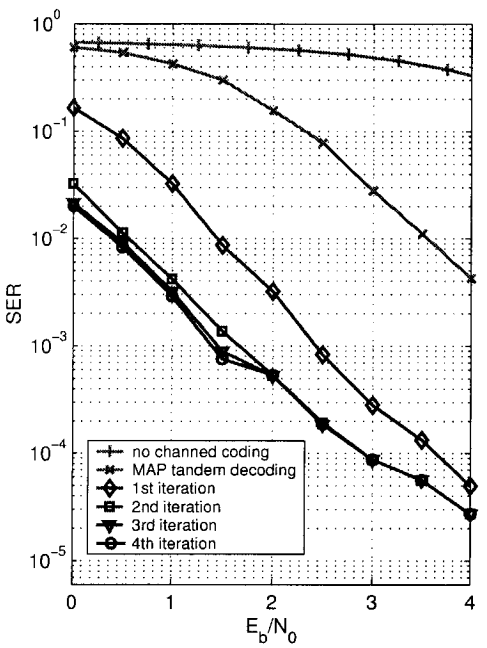
(a) BER for the proposed iterative scheme.



(b) SER for the proposed iterative scheme.



(c) BER for the iterative scheme of [20].



(d) SER for the iterative scheme of [20].

Fig. B.2 BER and SER for different E_b/N_0 (coded) and for both the proposed iterative scheme and the one in [20] . The joint decoder has knowledge of $N = 200$.

References

[1] T. M. Cover, "Elements of information theory", Wiley Series in Telecommunications, 1991.

[2] C. E. Shannon, "A mathematical theory of communication", *Bell Systems Technical Journal*, vol. 27, pp. 379–423, July-Oct. 1948.

[3] R. L. Dobrushin, "General formulation of Shannon's main theorem in information theory", *Amer. Math. Soc. Tmns.*, vol. 33, pp. 323–438, AMS, Providence, RI, 1963.

[4] S. Verdu, "A general formula for channel capacity", *IEEE Transactions on Information Theory*, vol. 40, no. 4, pp. 1147–1157, July 1994.

[5] J. L. Massey, "Joint source and channel coding", in *Communication Systems and Random Process Theory*, J. K. Skwirzynski, Ed. The Netherlands: Sojthoff and Nordhoff, pp. 279–293, 1978.

[6] S. Vembu, S. Verdu, Y. Steinberg, "The source-channel separation theorem revisited", *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 44–54, Jan. 1995.

[7] V. Cox, J. Hagenauer, N. Seshradi, C. E. W. Sundberg, "Subband speech coding and matched convolutional channel coding for mobile radio channels, *IEEE Transactions on Signal Processing"*, vol. 39, no. 8, pp. 1717–1731, Aug. 1991.

[8] K. Sayood, "Use of residual redundancy in the design of joint source/channel coders", *IEEE Transactions on Communications*, vol. 39, 6, pp. 838–846, June 1991.

[9] K. Sayood, Fuling Liu, J. D. Gibson, "A constrained joint source/channel coder design", *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 9, pp. 1584–1593, Dec. 1995.

[10] F. I. Alajaji, T. E. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech", *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 325–336, Sept. 1996.

[11] D. J. Miller, M. Park, "A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden Markov models", *IEEE Journal Transactions on Communications*, vol. 46, pp. 222–231, Feb. 1998.

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

[13] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes", *IEEE Transactions on Communications*, vol. 43, pp. 158–162, Apr. 1995.

[14] N. Phamdo, N. Farvardin, "Optimal detection of discrete Markov sources over discrete memoryless channels: Applications to combined source-channel coding", *IEEE Transactions on Information Theory*, vol. 40, pp. 186–193, Jan. 1994.

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

[16] J. G. Proakis, "Digital Communications", McGraw-Hill series in Electrical and Computer engineering, fourth edition, 2001.

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

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

[19] C. Berrou and A. Glavieux,"Near optimum error correcting coding and decoding: Turbo-codes", *IEEE Transactions on Communications*, vol. 44, pp. 1064–1070, Oct. 1996.

[20] A. Guyader, E. Fabre, C. Guillemot, M. Robert, "Joint source-channel turbo decoding of entropy-coded sources", *IEEE Journal on Selected Areas in Communications*, Vol. 19, no. 9, pp. 1680–1696, Sept. 2001.

[21] K. lakovic, J .Villasenor, "Combining variable length codes and turbo coded", *IEEE Vehicular Techonology Conference*, May 2002, pp. 1719–1723.

[22] J. Wen, J .Villasenor, "Soft-input soft-output decoding of variable length codes", *IEEE Transactions on Communications*, May 2002, pp. 688–6692.

[23] G. C. Zhu, F .Alajaji, J. Bajcsy, P. Mitran, "Transmission of nonuniform memoryless sources via nonsystematic turbo codes", *IEEE Transactions on Communications* Aug. 2004, pp. 1344–1354.

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

[25] T. Verma, J. Pearl, "Causal networks: semantics and expressiveness", *Uncertainty in Artificial Intelligence*, vol. 4, pp. 69–76, 1990.

[26] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, G. Cooper, "A Probabilistic Reformulation of the Quick Medical Reference System", *AAAI Spring Symposium Series AI in Medicine*, pp. 161–165, 1990.

[27] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann Publishers Inc., revised second printing, 1988.

[28] A. Guyader, "Contribution au algorithmes de décodage pour les codes graphices", PH.D Thesis, THÉSE Présentée devant l'Université de Rennes 1, No d'ordre: 2675, 2001.

[29] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles", *in Proc. Neural Inform. Processing Systems Conf.*, Dec. 1997.

[30] A. P. Dawid, "Conditional independence in statistical theory", *J.R. Stat. Society B*, 41, vol. 1, pp. 1–31, 1979.

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

[32] R. J. McEliece, D. J. C. MacKay, J.-F. Cheng,"Turbo decoding as an instance of Pearl's belief propagation algorithm", *IEEE Journal on Selected Areas in Communincations*, vol. 16, pp. 140–152, Feb. 1998

[33] B. Vucetic, J. Yuan, "Turbo Codes: Principles and Applications", Kluwer Academic Publishers, 2000.