Chapter 2

Trellis-based Detection Techniques

2.1 Introduction

In this chapter, we provide the reader with a brief introduction to the main detection techniques which will be relevant for the low-density parity-check (LDPC) coded modulation schemes considered in the following chapters of this book. In particular, we describe possible trellis-based detection techniques, taking also into account the presence of (binary) coding. Soft-output trellis based detection will be used as component block of LDPC coded modulation receivers in Chapters 5 and 7.

In Section 2.2, we provide a simple classification between hard-output and soft-output detection, presenting the Viterbi algorithm (VA) and the forward-backward (FB) algorithm as key examples of these two detection strategies, respectively. In Section 2.3, we quickly describe optimal detection strategies over channels with memory. In Section 2.4, we consider detection of encoded data. Section 2.5 concludes the chapter.

2.2 Hard-Output and Soft-Output Detection

A general model of a digital transmission system can be viewed as based on both a single M^K-ary signaling act or K repetitions of M-ary signaling acts. In the former interpretation, the message is the entire information sequence, whereas in the latter the message corresponds to an individual information symbol. According to these interpretations, two maximum a posteriori (MAP) detection strategies are obtained. MAP *sequence* detection is optimal in the

M. Franceschini, G. Ferrari and R. Raheli, *LDPC Coded Modulations,*
DOI: 10.1007/978-3-540-69457-1_2, © Springer-Verlag Berlin Heidelberg 2009

sense that it minimizes the probability of erroneously detecting the entire sequence, i.e., selecting a sequence not equal to the transmitted one. MAP *symbol* detection minimizes the probability of erroneously detecting each information symbol. More precisely, considering a suitable discretization process, through which the received signal is converted into an equivalent sequence of discrete-time observations $\boldsymbol{r}$, whose dimension depends on the number of samples per symbol interval, the following general formulations of these detection strategies can be derived:

$$\begin{aligned} \hat{\boldsymbol{a}} &= \underset{\boldsymbol{a}}{\operatorname{argmax}}\, P\{\boldsymbol{a}|\boldsymbol{r}\} && \text{MAP sequence detection} \\ \hat{a}_k &= \underset{a_k}{\operatorname{argmax}}\, P\{a_k|\boldsymbol{r}\} && \text{MAP symbol detection} \end{aligned}$$

where $\boldsymbol{a} \triangleq \{a_k\}_{k=0}^{K-1}$ and $\boldsymbol{r} \triangleq \{r_k\}_{k=0}^{K-1}$ are the sequences of K information symbols and observables, respectively.[1] As mentioned in the previous section, two algorithms that efficiently implement these two detection strategies are the VA [14,15] and the FB algorithm [17]. Both these algorithms are trellis-based, in the sense that they make use of a *trellis* diagram induced by a finite state machine (FSM) model of the underlying transmitter/channel model. More details will be provided in the remainder of this chapter.

The VA allows to efficiently derive the sequence of *hard* decisions $\hat{\boldsymbol{a}}$. On the opposite, the FB algorithm requires the computation of the a posteriori probability (APP) $P\{a_k|\boldsymbol{r}\}$ of each information symbol. In the case of binary information symbols, a commonly considered reliability value is given by the *logarithmic likelihood ratio* (LLR), derived from the APPs as follows:

$$\mathrm{LLR}_k \triangleq \log \frac{P\{a_k = 0|\boldsymbol{r}\}}{P\{a_k = 1|\boldsymbol{r}\}}. \tag{2.1}$$

It is immediate to recognize that a LLR captures, as a single quantity, the relationship between the APP of a transmitted "1" and that of a transmitted "0." Note that the formulation, based on the use of LLR, can also be extended to the case of larger information symbol cardinality. In the case of M-ary symbols, $(M-1)$ LLRs are needed: the LLR relative to the m-th symbol, $m = 0, \ldots, M-2$, is given by the logarithm of the ratio between $P\{a_k = m|\boldsymbol{r}\}$ and $P\{a_k = M-1|\boldsymbol{r}\}$—in other words, the reference probability is the APP of the last symbol, and its corresponding LLR is thus 0.

[1] Should channel coding or oversampling be used, each information symbol a_k could correspond to more than one observable. However, this can be straightforwardly taken into account by considering a vector notation, i.e., replacing the scalar r_k with a vector $\boldsymbol{r}_k$.

According to the discussion in the previous paragraph, rather than classifying trellis-based detection algorithms depending on the MAP detection strategy (either sequence or symbol), a more practical classification is based on the distinction between hard-output (VA) or soft-output (FB algorithm) detection.

2.2.1 The Viterbi Algorithm

As anticipated at the beginning of this section, the MAP sequence detection strategy can be formulated as

$$\widehat{\boldsymbol{a}} = \underset{\boldsymbol{a}}{\operatorname{argmax}}\, P\{\boldsymbol{a}|\boldsymbol{r}\}. \tag{2.2}$$

The operation in (2.2) can be described as "finding the sequence $\widehat{\boldsymbol{a}}$ such that the *a posteriori* probability $P\{\widehat{\boldsymbol{a}}|\boldsymbol{r}\}$ of having transmitted $\widehat{\boldsymbol{a}}$, given the received sequence $\boldsymbol{r}$, is maximum." In particular, a conceptual method for the identification of $\widehat{\boldsymbol{a}}$ consists of the evaluation of $P\{\boldsymbol{a}|\boldsymbol{r}\}$ for all possible information sequences $\boldsymbol{a}$, and selection of the sequence which maximizes the a posteriori probability. By chain factorization and owing to the independence of the information symbols, the MAP sequence detection strategy in (2.2) can be formulated in terms of the conditional probability density function (pdf) $p(\boldsymbol{r}|\boldsymbol{a})$ and the a priori sequence probability $P(\boldsymbol{a})$:

$$\begin{aligned} P\{\boldsymbol{a}|\boldsymbol{r}\} \sim p(\boldsymbol{r}|\boldsymbol{a})P(\boldsymbol{a}) &= \prod_{k=0}^{K-1} p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}) P\{a_k\} \\ &= \prod_{k=0}^{K-1} p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_0^{k}) P\{a_k\} \\ &\sim \sum_{k=0}^{K-1} \Big[\ln p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_0^{k}) + \ln P\{a_k\}\Big] \end{aligned} \tag{2.3}$$

where the symbol $\sim$ indicates that two quantities are monotonically related with respect to the variable of interest (in this case, $\boldsymbol{a}$); a statistical notion of system causality is assumed to hold in the second line; the monotonicity of the logarithm is used in the third line; and $\boldsymbol{r}_0^{k-1}$ is a short-hand notation for the sequence $\{r_i\}_{i=0}^{k-1}$ of k observables. In particular, if the a priori probabilities $\{P\{a_k\}\}$ are equal, the MAP sequence detection criterion coincides with the so-called maximum likelihood (ML) sequence detection criterion.

The maximization of (2.3) over all possible sequences $\{\boldsymbol{a}\}$ can be implemented as a search of a *path* in a *tree* diagram where each branch is in a

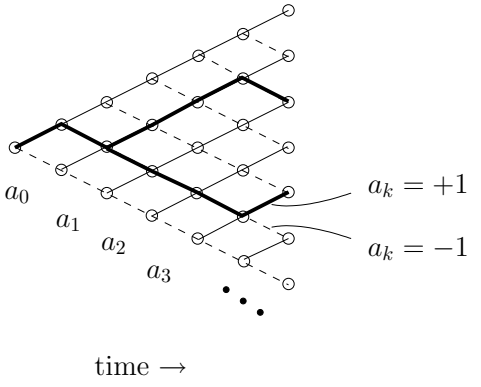

Figure 2.1: Tree representation of the possible transmitted sequences. The number of possible paths to be searched to perform MAP sequence detection increases exponentially with the sequence length.

one-to-one correspondence with an information symbol a_k. Consequently, a path is in a one-to-one correspondence with a *partial* sequence $\boldsymbol{a}_0^k$ up to epoch k. Assigning a *metric* equal to the k-th term of (2.3) to a branch at epoch k associated with symbol a_k and defining a path metric as the sum of the metrics of the branches forming the path, the MAP sequence detection strategy can be implemented as a search of the path with largest metric in this tree diagram. In Figure 2.1, an example of tree comprising all possible paths to be searched is shown, considering a binary alphabet for a_k. Two paths are also emphasized.

While a tree diagram is in principle required, the tree can often be "folded" into a *trellis*, as explained in the following. We assume that the encoder/modulator can be described as a finite state machine (FSM) with state μ_k and characterized by the following "next-state" and "output" functions, respectively:

$$\begin{cases} \mathrm{ns}(\mu_k, a_k) = \mu_{k+1} \\ \mathrm{o}(\mu_k, a_k) = c_k. \end{cases} \tag{2.4}$$

Considering a channel with complex additive white Gaussian noise (AWGN) with circular symmetry, i.e., with independent real and imaginary components, each with variance σ^2—a simple and very common example of memoryless channel—the generic observation at epoch k can be written as

$$r_k = c_k + n_k \tag{2.5}$$

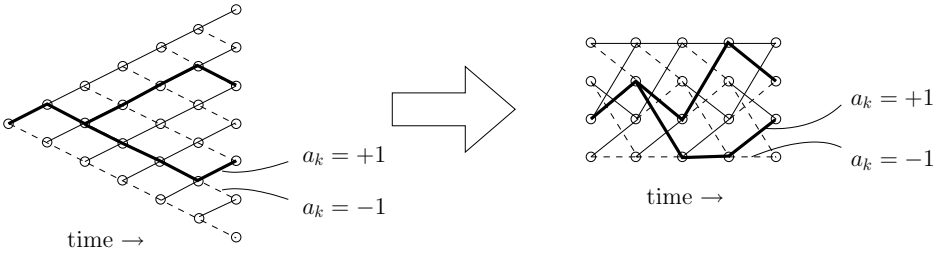

Figure 2.2: Illustrative representation of a tree folded into a trellis.

where n_k is the AWGN sample. In this case, it follows that

$$p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_0^k) = p(r_k|\mu_k, a_k) = \frac{1}{2\pi\sigma^2} e^{-\frac{|r_k - c_k|^2}{2\sigma^2}}$$

and (2.3) can be reformulated as

$$\begin{aligned} P\{\boldsymbol{a}|\boldsymbol{r}\} &\sim \sum_{k=0}^{K-1} \left[-\frac{|r_k - c_k|^2}{2\sigma^2} + \ln P\{a_k\} \right] \\ &\sim \sum_{k=0}^{K-1} \left[-|r_k - c_k|^2 + 2\sigma^2 \ln P\{a_k\} \right]. \end{aligned} \tag{2.6}$$

As mentioned above, a brute-force approach to the implementation of the MAP sequence detection criterion would consist in evaluating (2.6) for all possible information sequences, and choosing the maximizing sequence. Assuming M-ary information symbols, the complexity of this brute-force approach would be M^K, i.e., exponential with the transmission length. Hence, this implementation of the MAP sequence detection principle is feasible only for short transmission lengths, whereas it becomes impractical for transmission lengths of practical relevance in many applications. A much more efficient and appealing MAP sequence detection algorithm is the VA, which will be described in the following.

In the case of strictly finite-memory channels, MAP sequence detection can be formulated as indicated in (2.6), possibly by redefining the symbol c_k and the underlying FSM model. In particular, the optimal tree diagram can be folded into a *trellis diagram*, where the possible states at each epoch are given by all possible values of the state μ_k of the encoder/modulator FSM. In Figure 2.2, an illustrative representation of a folding of a tree into a trellis is shown. In the remainder of this chapter, the number of states of the en-

coder/modulator FSM will be indicated by S_c. Denoting by $t_k \triangleq (\mu_k, a_k)$ a transition in the trellis diagram, a *branch metric* associated with this transition can be defined as follows:

$$\lambda_k(\mu_k, a_k) = \lambda_k(t_k) \triangleq \ln p(r_k|\mu_k, a_k) + \ln P\{a_k\}. \tag{2.7}$$

Upon the definition of the branch metric (2.7), the a posteriori sequence probability can be written as follows:

$$P\{\boldsymbol{a}|\boldsymbol{r}\} \sim \sum_{k=0}^{K-1} \lambda_k(\mu_k, a_k). \tag{2.8}$$

Without entering into the details (the interested reader can find plenty of literature regarding the VA, e.g. [4, 21]), the implementation principle of the VA is that of associating to each state μ_n a *partial path metric* relative to the corresponding path originating from a known state μ_0, at epoch 0, and terminating into μ_n. This partial path metric, denoted as $\Lambda_n(\mu_n)$, can be written as follows:

$$\Lambda_n(\mu_n) = \sum_{k=0}^{n-1} \lambda_k(t_k) \qquad 1 \leq n \leq K. \tag{2.9}$$

Obviously, $P\{\boldsymbol{a}|\boldsymbol{r}\} = \Lambda_K(\mu_K)$. Based on the trellis representation of the underlying FSM, the partial metrics, associated with the trellis states, can be computed recursively. For the sake of simplicity, we consider binary information symbols, i.e., $M = 2$. The path metrics associated to states $\mu_k^{(1)}$ and $\mu_k^{(2)}$ are indicated as $\Lambda_k(\mu_k^{(1)})$ and $\Lambda_k(\mu_k^{(2)})$, respectively. The VA associates to state μ_{k+1} (the common ending state of both transitions $t_k^{(1)} = (\mu_k^{(1)}, a_k^{(1)})$ and $t_k^{(2)} = (\mu_k^{(2)}, a_k^{(2)})$) the following path metric:

$$\Lambda_{k+1}(\mu_{k+1}) = \max\left\{\Lambda_k(\mu_k^{(1)}) + \lambda_k(\mu_k^{(1)}, a_k^{(1)}), \Lambda_k(\mu_k^{(2)}) + \lambda_k(\mu_k^{(2)}, a_k^{(2)})\right\}. \tag{2.10}$$

In this sense, the basic operation of the VA is defined as *add-compare-select* (ACS), since: (i) the path metrics associated with the two starting states are summed with the branch metrics of the two branches entering into the common final state; (ii) the obtained partial path metrics are compared; and (iii) the "candidate" largest path metric is selected as *the* path metric associated to μ_{k+1}.

The evaluation of path metrics as indicated above guarantees that the path terminating into any state at a given epoch is, among all entering paths, the one to which the largest metric is associated. At any epoch, the S_c paths with the largest possible path metrics are therefore tracked. Consequently, at the final trellis section at epoch K, the largest path metric among those associated with the final states is such that the corresponding information sequence satisfies the MAP sequence detection criterion in (2.2).

Even if the complexity of the VA is proportional to KS_c (i.e., the dependence on the transmission length is linear and not exponential anymore), the delay would still be unacceptable for large transmission lengths, since one should wait for the transmission of the entire information sequence before being able to output the sequence of symbols satisfying the MAP sequence detection criterion. The VA, however, has an appealing feature which we will describe shortly. At every epoch the number of survivors is equal to the number S_c of states in the trellis. One could track backwards all these survivors starting from the trellis section at epoch k down to the trellis section at epoch $k - D$. For each value of D it is possible to count the number of distinct survivor paths. This number is a monotonically non-increasing function of D. In particular, if the value of D is such that the number of distinct paths is equal to 1, all S_c survivors share the same path from trellis section 0 to trellis section $k - D$. In other words, the survivors merge at section $k - D$. This implies that there is no uncertainty on the best sequence from epoch 0 to epoch $k - D$, and, therefore, the corresponding decisions can be emitted. The probability of having all survivors sharing, at epoch k, the same path in correspondence to trellis section $k - D$ rapidly approaches 1 for increasing values of D. Hence, at epoch k, it is possible to emit the decision $\widehat{a}_{k-D}$ relative to the MAP information sequence which will eventually be selected. In other words, by assuming that consecutive observations are sequentially available, the latency corresponds to only D symbol intervals, where D is sufficiently large to guarantee, at epoch k, high probability of merged survivors at section $k - D$.

2.2.2 The Forward-Backward Algorithm

The most commonly used algorithm to perform MAP symbol detection is the so-called FB algorithm. Seminal work on the design of algorithms for soft-output decoding dates back to the late Sixties [22, 23]. An instance of the FB algorithm was proposed in [24], but a clear formalization is due to Bahl, Cocke, Jelinek and Raviv in 1974 [17]—for this reason, the FB algorithm is also often referred to as BCJR algorithm from the initials of the last names of

the authors of [17].

The MAP symbol detection criterion leads to the choice of the symbol $\hat{a}_k$ which minimizes the probability of error with respect to the received signal. More precisely, the MAP symbol detection strategy can be formulated as follows:

$$\hat{a}_k = \underset{a_k}{\operatorname{argmax}} P\{a_k|\boldsymbol{r}\}. \tag{2.11}$$

In order to compute the APP $P\{a_k|\boldsymbol{r}\}$ one can write:

$$P\{a_k|\boldsymbol{r}\} \quad = \quad \sum_{\boldsymbol{a}:a_k} P\{\boldsymbol{a}|\boldsymbol{r}\} \sim \sum_{\boldsymbol{a}:a_k} p(\boldsymbol{r}|\boldsymbol{a})P\{\boldsymbol{a}\} \tag{2.12}$$

where the notation $\boldsymbol{a} : a_k$ denotes all possible information sequences containing a_k, or *compatible* with a_k. Note that the computation of the APP, as indicated in (2.12), can be based on the same metric as in the case of MAP sequence detection (i.e., $p(\boldsymbol{r}|\boldsymbol{a})P\{\boldsymbol{a}\}$) and a further *marginalization* based on the sum over all the information sequences compatible with symbol a_k. Assuming that the information symbols are independent, one can further express the APP as follows:

$$P\{a_k|\boldsymbol{r}\} \quad \sim \quad P\{a_k\} \sum_{\boldsymbol{a}:a_k} p(\boldsymbol{r}|\boldsymbol{a}) \prod_{i=0, i\neq k}^{K-1} P\{a_i\}. \tag{2.13}$$

The first and simplest possible approach for the evaluation of the APP could be based on the computation of expression (2.13). It is immediate to conclude that the computational efficiency of this operation is very low, since one must compute a large number of sequence metrics and then marginalize by adding them together. The complexity would obviously be exponential in the transmission length K. The FB algorithm, introduced in more detail in the following, represents an efficient way to compute the APP, with a complexity linear with the transmission length K, as in the case of a VA for MAP sequence detection.

As already mentioned, the first clear formulation of the FB algorithm can be found in [17]. In the following, we propose a simple derivation of the FB algorithm for transmission over a *memoryless channel*. We assume that the encoder/modulator can be represented as a FSM, with state μ_k and output symbol c_k. We assume that the next-state and the output functions are known[2]

[2]Note that the derivation shown in the following holds also in the case of a channel with strictly finite-memory, the only difference being the interpretation of μ_k as the state

and can be formulated as in (2.4). The couple (μ_k, a_k) uniquely identifies a transition in the trellis diagram of the encoder/modulator FSM. We denote this transition as t_k. After a suitable discretization process with one sample per information symbol, we assume that the observable at epoch k can be written as in (2.5). In this case, the APP can be expressed as follows:

$$\begin{aligned} P\{a_k|\boldsymbol{r}\} &\sim p(\boldsymbol{r}|a_k)P\{a_k\} \\ &= \sum_{\mu_k} p(\boldsymbol{r}|a_k,\mu_k)P\{\mu_k|a_k\}P\{a_k\} \\ &= \sum_{\mu_k} p(\boldsymbol{r}_{k+1}^{K-1}|\boldsymbol{r}_0^k,a_k,\mu_k)p(r_k|\boldsymbol{r}_0^{k-1},a_k,\mu_k)p(\boldsymbol{r}_0^{k-1}|a_k,\mu_k) \\ &\qquad \cdot P\{\mu_k|a_k\}P\{a_k\}. \end{aligned} \tag{2.14}$$

Assuming independent information symbols, since μ_k may depend on $\boldsymbol{a}_0^{k-1}$, it follows that:

$$P\{\mu_k|a_k\} = P\{\mu_k\}.$$

Upon the assumption of transmission over a memoryless channel, the remaining conditional pdfs in (2.14) can be simplified as:

$$\begin{aligned} p(\boldsymbol{r}_{k+1}^{K-1}|\boldsymbol{r}_0^k,a_k,\mu_k) &= p\big(\boldsymbol{r}_{k+1}^{K-1}|\mu_{k+1}=\mathrm{ns}(a_k,\mu_k)\big) \\ p(r_k|\boldsymbol{r}_0^{k-1},a_k,\mu_k) &= p(r_k|a_k,\mu_k) \\ p(\boldsymbol{r}_0^{k-1}|a_k,\mu_k) &= p(\boldsymbol{r}_0^{k-1}|\mu_k). \end{aligned}$$

Hence, the APP in (2.14) can be rewritten as follows:

$$\begin{aligned} P\{a_k|\boldsymbol{r}\} \sim \sum_{\mu_k} & p\big(\boldsymbol{r}_{k+1}^{K-1}|\mu_{k+1}=\mathrm{ns}(a_k,\mu_k)\big) \\ & \cdot p(r_k|a_k,\mu_k)p(\boldsymbol{r}_0^{k-1}|\mu_k)P\{\mu_k\}P\{a_k\}. \end{aligned} \tag{2.15}$$

By defining

$$\begin{aligned} \alpha_k(\mu_k) &\triangleq p(\boldsymbol{r}_0^{k-1}|\mu_k)P\{\mu_k\} \\ \gamma_k(a_k,\mu_k) &\triangleq p(r_k|a_k,\mu_k)P\{a_k\} \\ \beta_{k+1}(\mu_{k+1}) &\triangleq p(\boldsymbol{r}_{k+1}^{K-1}|\mu_{k+1}) \end{aligned}$$

of the FSM obtained by concatenating the encoder/modulator and the channel. Moreover, we assume generation of a single output symbol c_k in correspondence to each information symbol a_k, but the derivation can be straightforwardly extended to the case of multiple output symbols by using a vector notation, as mentioned in Footnote 1.

the desired symbol APP finally becomes

$$P\{a_k|\boldsymbol{r}\} \sim \sum_{\mu_k} \alpha_k(\mu_k)\gamma_k(a_k,\mu_k)\beta_{k+1}(\mu_{k+1}) \tag{2.16}$$

where, for the sake of notational simplicity, the dependence of μ_{k+1} on μ_k and a_k is not explicitly indicated.[3] In the following, since the generated soft-output value is a quantity monotonically related with the APP, we will indicate this value with the general notation $\mathrm{S}[a_k]$. In other words, one can write:

$$\mathrm{S}[a_k] \triangleq \sum_{\mu_k} \alpha_k(\mu_k)\gamma_k(a_k,\mu_k)\beta_{k+1}(\mu_{k+1}). \tag{2.17}$$

The actual APP values can be obtained by applying a proper normalization to the terms $\mathrm{S}[a_k]$, since $\sum_{a_k} P\{a_k|\boldsymbol{r}\} = 1$. Note that the operation (2.17) where the quantities $\{\alpha_k(\mu_k)\}$ and $\{\beta_{k+1}(\mu_{k+1})\}$ are combined to generate the APP is usually referred to as *completion*.

The quantities $\alpha_k(\mu_k)$ and $\beta_{k+1}(\mu_{k+1})$ can be computed by means of forward and backward recursions, respectively. More precisely, one can write:

$$\begin{aligned}
\alpha_k(\mu_k) &= p(\boldsymbol{r}_0^{k-1}|\mu_k)P\{\mu_k\} \\
&= \sum_{\substack{(\mu_{k-1},a_{k-1}):\\ \mathrm{ns}(a_{k-1},\mu_{k-1})=\mu_k}} p(\boldsymbol{r}_0^{k-1}|a_{k-1},\mu_{k-1},\mu_k)P\{a_{k-1},\mu_{k-1}|\mu_k\}P\{\mu_k\} \\
&= \sum_{\substack{(\mu_{k-1},a_{k-1}):\\ \mathrm{ns}(a_{k-1},\mu_{k-1})=\mu_k}} p(r_{k-1}|\boldsymbol{r}_0^{k-2},a_{k-1},\mu_{k-1},\mu_k) \\
&\qquad \cdot p(\boldsymbol{r}_0^{k-2}|a_{k-1},\mu_{k-1},\mu_k)P\{a_{k-1},\mu_{k-1},\mu_k\} \\
&= \sum_{\substack{(\mu_{k-1},a_{k-1}):\\ \mathrm{ns}(a_{k-1},\mu_{k-1})=\mu_k}} p(r_{k-1}|\boldsymbol{r}_0^{k-2},a_{k-1},\mu_{k-1},\mu_k) \\
&\qquad \cdot p(\boldsymbol{r}_0^{k-2}|a_{k-1},\mu_{k-1},\mu_k)P\{\mu_k|a_{k-1},\mu_{k-1}\} \\
&\qquad \cdot P\{a_{k-1}|\mu_{k-1}\}P\{\mu_{k-1}\}
\end{aligned} \tag{2.18}$$

where the index of the summation indicates all possible transitions $\{(\mu_{k-1}, a_{k-1})\}$ compatible, through the next-state function, with μ_k—this notation is general and accounts also for the case of underlying recursive and non-recursive

[3] This simplifying notational assumption (and other similar assumptions) will also be used in the following. The context should eliminate any ambiguity.

FSM models. The summation in (2.18) can be re-interpreted as carried over all possible trellis branches $\{t_{k-1}\}$ compatible with the final state μ_k. Since we are considering possible combinations of μ_{k-1} and a_{k-1} compatible with μ_k, it follows that

$$P\{\mu_k|a_{k-1},\mu_{k-1}\}=1. \tag{2.19}$$

On the basis of the independence between the information symbols and recalling the absence of channel memory, one can write:

$$\begin{aligned} p(r_{k-1}|\boldsymbol{r}_0^{k-2},a_{k-1},\mu_{k-1},\mu_k) &= p(r_{k-1}|\mu_{k-1},a_{k-1}) \\ p(\boldsymbol{r}_0^{k-2}|a_{k-1},\mu_{k-1},\mu_k) &= p(\boldsymbol{r}_0^{k-2}|\mu_{k-1}) \\ P\{a_{k-1}|\mu_{k-1}\} &= P\{a_{k-1}\}. \end{aligned}$$

Finally, a step in the forward recursion in (2.18) can be concisely expressed as follows:

$$\begin{aligned} \alpha_k(\mu_k) &= \sum_{t_{k-1}:\mu_k} p(\boldsymbol{r}_0^{k-2}|\mu_{k-1})P\{\mu_{k-1}\}p(r_{k-1}|\mu_{k-1},a_{k-1})P\{a_{k-1}\} \\ &= \sum_{t_{k-1}:\mu_k} \alpha_{k-1}(\mu_{k-1})\gamma_{k-1}(t_{k-1}). \end{aligned} \tag{2.20}$$

A similar derivation holds also for the backward recursion. More precisely, one can write:

$$\begin{aligned} \beta_k(\mu_k) &= p(\boldsymbol{r}_k^{K-1}|\mu_k) \\ &= \sum_{a_k} p(\boldsymbol{r}_k^{K-1}|a_k,\mu_k)P\{a_k|\mu_k\} \\ &= \sum_{a_k} p(r_k|\boldsymbol{r}_{k+1}^{K-1},a_k,\mu_k)p(\boldsymbol{r}_{k-1}^{K-1}|a_k,\mu_k)P\{a_k|\mu_k\}. \end{aligned} \tag{2.21}$$

On the basis of the independence between information symbols and the absence of memory of the considered transmission channel, the following simplifications hold:

$$\begin{aligned} p(r_k|\boldsymbol{r}_{k+1}^{K-1},a_k,\mu_k) &= p(r_k|a_k,\mu_k) \\ p(\boldsymbol{r}_{k-1}^{K-1}|a_k,\mu_k) &= p\big(\boldsymbol{r}_{k-1}^{K-1}|\mu_{k+1}=\mathrm{ns}(a_k,\mu_k)\big) \\ P\{a_k|\mu_k\} &= P\{a_k\}. \end{aligned}$$

A step in the backward recursion, as indicated in (2.21), can be rewritten as follows:

$$\begin{aligned}\beta_k(\mu_k) &= \sum_{a_k} p(\boldsymbol{r}_{k-1}^{K-1}|\mu_{k+1})p(r_k|a_k,\mu_k)P\{a_k\} \\ &= \sum_{a_k} \beta_{k+1}(\mu_{k+1})\gamma_k(t_k).\end{aligned}$$

2.3 Optimal Detection Strategies for Channels with Memory

In this section, we recall the basic detection strategies which can be devised for communication channels with memory. In particular, as seen in the previous sections, the same basic *metric* can be used for both hard-output and soft-output detection. For more details, the reader is referred to [25, 26].

Consider the transmission system model previously described. A sequence of independent and identically distributed M-ary information symbols $\{a_k\}$ are transmitted successively from epoch 0 to epoch $K-1$. The encoder/modulator block can be described as a time-invariant FSM (e.g., a trellis coded modulator, TCM, [27] or a continuous phase modulator, CPM, [28]), and we assume that next-state and output functions can be expressed as in (2.4).

A *causality condition* for the considered communication system can be formulated in terms of statistical dependence of the observation sequence $\boldsymbol{r}_0^k$, up to epoch k, on the information sequence. Accordingly, a system is causal if

$$p(\boldsymbol{r}_0^k|\boldsymbol{a}) = p(\boldsymbol{r}_0^k|\boldsymbol{a}_0^k). \tag{2.22}$$

Similarly, a *finite-memory condition* (FMC) can be formulated, in statistical terms, as follows:

$$p(r_k|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_0^k) = p(r_k|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_{k-C}^k,\mu_{k-C}) \tag{2.23}$$

where C is a suitable *finite-memory parameter* and μ_{k-C} represents the state, at epoch $k-C$, of the encoder/modulator. The considered model includes any definition of state μ_k in terms of a suitable state variable, not necessarily defined in terms of input variables. It can easily be proved that causality and finite-memory conditions imply the following equalities [26]:

$$p(r_k|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_{k-D}^k,\mu_{k-D}) = p(r_k|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_{k-C}^k,\mu_{k-C}) \qquad \forall D \geq C \tag{2.24}$$

$$p(\boldsymbol{r}_k^{K-1}|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_0^{k-1}) = p(\boldsymbol{r}_k^{K-1}|\boldsymbol{r}_0^{k-1},\boldsymbol{a}_{k-C}^{k-1},\mu_{k-C}). \tag{2.25}$$

The first equality formalizes the intuition that considering past information symbols, before epoch $k - C$, adds no further information regarding the observation at epoch k. The second equality formalizes the idea that the finite-memory condition[4] extends to future observations beyond epoch k. Upon the introduction of an output function $\mathrm{o}(\cdot, \cdot)$ as in (2.4), causality and finite-memory conditions can be formulated as follows:

$$p(\boldsymbol{r}_0^k|\boldsymbol{c}) = p(\boldsymbol{r}_0^k|\boldsymbol{c}_0^k) \tag{2.26}$$

$$p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{c}_0^k) = p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{c}_{k-C}^k). \tag{2.27}$$

We remark, however, that these conditions involve the transmission channel only and, in general, *do not* imply (2.22) and (2.23). A case of interest may be that of a linear block code followed by a memoryless modulator. In particular, a linear block code is not guaranteed to be causal and finite-memory[5] so that the channel causality (2.26) and finite memory (2.27) do not imply the system causality (2.22) and finite memory condition (2.23).

The introduction of the FMC leads naturally to the definition of augmented trellis state and branch (transition):

$$\begin{aligned} S_k &\triangleq (\boldsymbol{a}_{k-C}^{k-1}, \mu_{k-C}) \\ T_k &\triangleq (S_k, a_k) = (\boldsymbol{a}_{k-C}^{k}, \mu_{k-C}). \end{aligned}$$

Then, the following common metric can be used in the VA and FB algorithm:

$$\gamma_k(T_k) \triangleq p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_{k-C}^k, \mu_{k-C}) P\{a_k\}. \tag{2.28}$$

In particular:

- the VA can now be formulated in the logarithmic domain, by defining the branch metric $\lambda_k(T_k) \triangleq \log \gamma_k(T_k)$, and obtaining

$$\log P\{\boldsymbol{a}|\boldsymbol{r}\} \sim \sum_{k=0}^{K-1} \lambda_k(T_k);$$

- the symbol APP computed by the FB algorithm can be finally expressed as

$$P\{a_k|\boldsymbol{r}\} \sim \sum_{S_k} \beta_{k+1}(\mathrm{NS}(S_k, a_k)) \gamma_k(S_k, a_k) \alpha_k(S_k).$$

[4]Note that there is a slight difference between the formal definition of the finite memory condition (2.23) and (2.25), since in (2.25) the conditioning information sequence is $\boldsymbol{a}_0^{k-1}$ and does not include symbol a_k. This is, however, expedient for the derivation of the backward recursion of the FB algorithm in Subsection 2.2.2.

[5]*Block-wise* causality and finite-memory must be indeed satisfied.

2.4 Detection of Encoded Data

In this section, we discuss on the applicability of the previously devised strategies in the case of encoded data. The presence of coding makes consecutive transmitted symbols *dependent*. In the case of transmission over channels with memory, it follows that the overall memory is due to the channel *and* the encoder. Depending on how these two memory components are "treated," detection and decoding can be properly combined or separated.

2.4.1 Joint Detection and Decoding

In the case of a channel characterized by parameters affected by stochastic uncertainty, a very general parametric model for the observation r_k is the following:

$$r_k = g(\boldsymbol{a}_{k-L}^k, \mu_{k-L}, \boldsymbol{\xi}_0^k) + w_k \tag{2.29}$$

where L is an integer quantifying the encoding memory (e.g., the memory length of a convolutional encoder), $\boldsymbol{\xi}_0^k$ is a sequence of stochastic parameters independent of $\boldsymbol{a}$, and w_k is an additive noise sample. Under this channel model, the following *conditional Markov property*

$$p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_0^k) = p(r_k|\boldsymbol{r}_{k-N}^{k-1}, \boldsymbol{a}_0^k) \tag{2.30}$$

where N is the order of Markovianity, is sufficient to guarantee a FMC. In fact, (2.30) implies the following [26]:

$$p(r_k|\boldsymbol{r}_0^{k-1}, \boldsymbol{a}_0^k) = p(r_k|\boldsymbol{r}_{k-N}^{k-1}, \boldsymbol{a}_{k-C}^k, \mu_{k-C}) \tag{2.31}$$

where the finite-memory parameter is $C = N + L$. It is immediate to recognize that (2.31) represents a special case of (2.23). As a consequence, all the derivations in the previous section hold by using the "exponential metric[6]" $\gamma_k(T_k) = p(r_k|\boldsymbol{r}_{k-N}^{k-1}, T_k)P\{a_k\}$. In other words, (2.31) is the key relation which "links" the algorithms derived in Section 2.3 with the detection problem over channels with memory. A statistical description of the stochastic channel parameter allows one to compute this exponential metric as [25, 26]

$$\begin{aligned} \gamma_k(T_k) &= \frac{p(\boldsymbol{r}_{k-N}^k|T_k)}{p(\boldsymbol{r}_{k-N}^{k-1}|S_k)} P\{a_k\} \\ &= \frac{\mathrm{E}_{\boldsymbol{\xi}_0^k}\{p(\boldsymbol{r}_{k-N}^k|T_k, \boldsymbol{\xi}_0^k)\}}{\mathrm{E}_{\boldsymbol{\xi}_0^{k-1}}\{p(\boldsymbol{r}_{k-N}^{k-1}|S_k, \boldsymbol{\xi}_0^{k-1})\}} P\{a_k\}. \end{aligned} \tag{2.32}$$

[6]The usual notation in the literature refers to a *metric* in the logarithmic domain. Hence, assuming that $\log \gamma_k$ can be referred to as metric, we refer to γ_k as *exponential metric*.

2.4.2 Separate Detection and Decoding

While in the previous subsection the fundamental metric γ was computed by taking into account simultaneously encoding and channel memories, another possible strategy consists in separating the detection process from the decoding process. This can be carried out, at the receiver side, by considering the concatenation of two blocks:

- the first block, i.e., the detector, computes a posteriori reliability values on the coded symbols, by taking into account the channel memory and exploiting the available statistical channel characterization;
- the second block acts as a "standard" decoder, which receives at its input the reliability values generated by the detector. However, care has to be taken in correctly estimating the distribution of the reliability values generated by the detector (typically, their distribution is well approximated as Gaussian [8]).

In the presence of a stochastic channel with order of Markovianity equal to N, as in Subsection 2.4.1, the detector can make use of the final metric (2.32), the only difference, with respect to Subsection 2.4.1, consisting of the fact that $L = 0$, i.e., $C = N$.

2.4.3 Iterative Detection and Decoding

As briefly anticipated in Chapter 1, the concept of *iterative decoding* was originally introduced by Gallager in his Ph.D. thesis [29] for decoding LDPC codes and was crystallized by Berrou and Glavieux in 1993 with the introduction of turbo codes [7, 8]. In this revolutionary work, the authors showed that a complicated code, with a particular structure, can be decoded efficiently with limited complexity. In particular, they considered a parallel concatenated convolutional code (PCCC), constituted by the parallel concatenation, through interleaving, of two convolutional codes. The receiver is based on two component decoders (corresponding to the two constituent convolutional encoders) which *exchange information* between each other. In Figure 2.3, the basic structure of a turbo decoder, relative to a PCCC, is shown. The two component decoders exchange *soft information* between each other and the interleaver is denoted as Π. More precisely, the decoders exchange a modified version of the APP, the so-called *extrinsic information*, which represents the component of the generated soft output on a symbol not depending on the soft-input information on the same symbol [8]. Referring to the FB algorithm

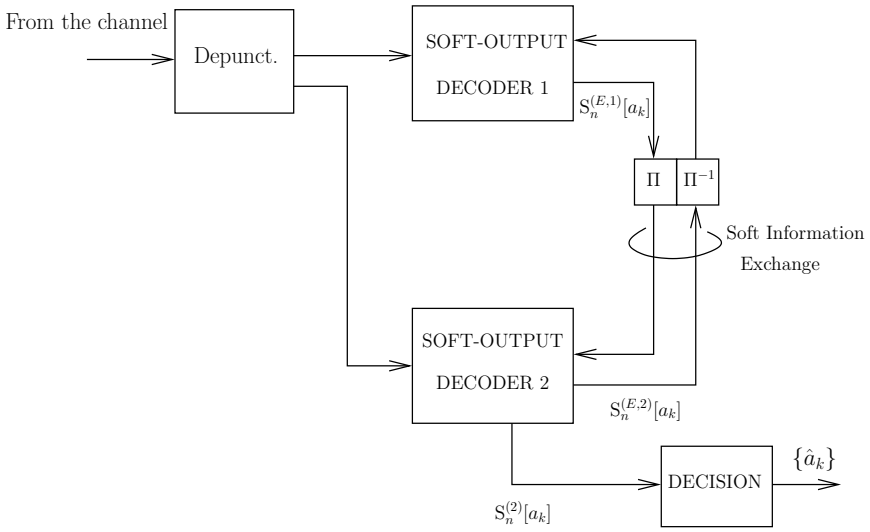

Figure 2.3: Turbo decoder for a PCCC.

formulation proposed in Subsection 2.2.2, the final soft-output quantity, i.e., the extrinsic information, (2.17), can be written as follows:

$$\mathrm{S}[a_k] = P\{a_k\}\mathrm{S}^{(E)}[a_k]$$

where

$$\mathrm{S}^{(E)}[a_k] \triangleq \sum_{\mu_k} \alpha_k(\mu_k)\gamma'_k(a_k,\mu_k)\beta_{k+1}(\mu_{k+1})$$

and

$$\gamma'_k(a_k,\mu_k) \triangleq p(r_k|a_k,\mu_k) = \frac{\gamma_k(a_k,\mu_k)}{P\{a_k\}}.$$

Note that the a priori probability $P\{a_k\}$ of an information symbol used by each component decoder is given, in the iterative decoding process, by the extrinsic information generated by the other decoder, i.e.,

$$P\{a_k\} \longleftarrow \mathrm{S}^{(E,i)}[a_k] = \frac{\sum_{\mu_k} \alpha_k(\mu_k)\gamma_k(a_k,\mu_k)\beta_{k+1}(\mu_{k+1})}{\mathrm{S}^{(E,j)}[a_k]} \tag{2.33}$$

where $i,j \in \{1,2\}$, $i \neq j$, and the operator "$\longleftarrow$" denotes a value assignment.

In Figure 2.3, the extrinsic information values on information symbol a_k generated by the first and second decoders at the n-th iteration are denoted by $\mathrm{S}_n^{(E,1)}[a_k]$ and $\mathrm{S}_n^{(E,2)}[a_k]$, respectively. Assuming, as a convention, that an iteration is constituted by the sequence of decoding acts of the first and second component decoders, the soft-output values generated by each component decoder can be re-written as follows:

$$\underbrace{\mathrm{S}_n^{(1)}[a_k]}_{\text{complete output}} = \underbrace{\mathrm{S}_{n-1}^{(E,2)}[a_k]}_{\text{input}} \underbrace{\mathrm{S}_n^{(E,1)}[a_k]}_{\text{extrinsic output}} \tag{2.34}$$

$$\mathrm{S}_n^{(2)}[a_k] = \mathrm{S}_n^{(E,1)}[a_k]\ \mathrm{S}_n^{(E,2)}[a_k]. \tag{2.35}$$

In other words, the soft-output value generated by the first decoder at the n-th iteration is the product between the soft value at its input, corresponding to the extrinsic information generated by the second decoder at the $(n-1)$-th iteration, and the generated extrinsic information. The soft-output value generated by the second decoder at the n-th iteration is the product between the soft value at its input, corresponding to the extrinsic information generated by the first decoder at the same iteration, and the generated extrinsic information. The two soft-output values in (2.34) and (2.35) indicate clearly that the soft-output decoding processes (based on the FB algorithm) in the two component decoders are *coupled*. If the iterative decoding process starts with decoder 1, then the soft-output information (2.35) produced by decoder 2 will be the final soft information at the output.

In the presence of channels with memory, this iterative decoding scheme can be straightforwardly extended to an iterative joint detection/decoding scheme, the only difference being the fact that each component decoder in Figure 2.3 makes use of an FB algorithm based on the metric developed in Subsection 2.4.1.

2.4.4 Turbo Detection

In the previous subsection we have considered iterative joint detection/decoding, where every component block was aware of both channel and code structure. A separate detection/decoding approach, obtained through a serial concatenation of a detector and decoder (as described in Subsection 2.4.2), leads to an iterative receiver scheme where the detector (or soft demodulator), which accounts for the statistical characterization of the channel, and the decoder, which accounts for the code structure, exchange soft information. More precisely, in the non-iterative separate scheme introduced in Subsection 2.4.2, the inner detector computes and passes soft-output information to the outer

decoder without a feedback from the decoder itself. The receiver becomes iterative as soon as the outer decoder feeds soft information back to the detector. In the literature, this iterative separate detection and decoding scheme is usually referred to as *turbo-equalization* [30], despite this terminology is slightly abused because, strictly speaking, an equalization process does not take place. An alternative terminology is "turbo detection."

2.5 Concluding Remarks

In this chapter, we have summarized basic detection theoretic concepts which will be used in the following chapters. In particular, we have discussed a unified approach to trellis-based detection over channels with memory. The concepts of joint and separate detection/decoding have been presented, together with a short discussion on iterative detection and turbo equalization. The FB algorithm—or other trellis-based soft-output detection algorithms derived from it, such as, for example, the multi-trellis soft-input soft-output (SISO) algorithm which will be presented in Section 7.6—will be used in the remainder of this book as component blocks to build LDPC coded modulation receivers.

http://www.springer.com/978-3-540-69455-7

LDPC Coded Modulations

Franceschini, M.; Ferrari, G.; Raheli, R.

2009, IX, 197 p., Hardcover
ISBN: 978-3-540-69455-7