COURS CNAM ELE203

Codes concaténés et Décodage itératif

Version 3.0

13 mai 2007

Didier Le Ruyet

Glossaire

APP : probabilité *a posteriori*, *a posteriori probability* en anglais
APRI : probabilité *a priori*, *a priori probability* en anglais
BBAG : bruit blanc additif gaussien
EXTR : probabilité extrinsèque
IRWEF : fonction d'énumération de poids, *input redundancy weight enumerator function* en anglais
code LDPC : code à matrice de parité de faible densité, *low density parity check* en anglais
LLR : logarithme du rapport de vraisemblance, *logarithm likelihood ratio* en anglais
MAP : *maximum a posteriori*
ML : *maximum likelihood*
codeur NRSC : codeur convolutif systématique non récursif, *non recursive systematic convolutional code* en anglais
codes PCC : codes convolutifs concaténés en parallèle, *parallel concatenated convolutional codes* en anglais
code RA : code à répétition accumulation, *repeat accumulated code* en anglais
codeur RSC : codeur convolutif récursif systématique, *recursive systematic convolutional code* en anglais
SERF : séquence d'entrée à réponse finie
SOVA : décodeur de Viterbi à sorties pondérées (*soft output viterbi algorithm* en anglais)
SNR : rapport signal à bruit, *signal to noise ratio* en anglais
BER : taux d'erreurs bit
WEF fonction d'énumération de poids *weight enumerator function*

Chapitre 1

Codes concaténés et décodage itératif

1.1 Introduction

Dans les années qui ont suivi les travaux de Shannon en 1948 [44], les chercheurs ont proposé des codes correcteurs d'erreurs de complexité raisonnable tout en ayant de bonnes distances minimales. L'idée de combiner plusieurs codes simples pour obtenir des codes plus performants a été proposée en 1954 avec les codes produits par Elias [15]. Nous verrons que cette idée permet d'obtenir des codes ayant des caractéristiques très proches de celles des codes aléatoires malgré une structure de codage simple.

Cependant pour ces codes, le décodage était fortement sous optimal car il faisait appel à des décodeurs à décisions fermes.

La découverte en 1993 par Berrou et Glavieux [9] des Turbo codes ou codes convolutifs concaténés en parallèle et la redécouverte des codes LDPC (*low density parity check code* en anglais) a enfin permis de se rapprocher nettement de la limite de capacité (quelques dizièmes de dB pour des trames de plusieurs milliers de bits sur un canal BBAG).

Pour le décodage de ces codes les auteurs on proposé d'utiliser un décodeur itératif dont les décodeurs élémentaires sont à entrées et sorties pondérées. Bien que le décodage itératif soit sous optimal il converge presque toujours vers la solution obtenue par le décodage à maximum de vraisemblance.

En résumé, les trois principaux points fondamentaux que nous étudierons dans ce chapitre sont les suivants :

- l'approche du codage aléatoire obtenue grâce à la concaténation de plusieurs codes et l'utilisation d'entrelaceurs,
- le remplacement du décodage à sorties fermes des codes constituants par le décodage à sorties pondérées,
- l'introduction du décodage itératif.

Nous commencerons par introduire les techniques de décodage à entrées et sorties pondérées. Ensuite nous présenterons les codes PCBC, LDPC. Finalement nous étudierons les codes convolutifs concaténés en parallèle ou Turbo codes ainsi que les codes convolutifs concaténés en série.

1.2 Décodage à entrées et sorties pondérées

1.2.1 Introduction

Dans le chapitre précédent, nous avons introduit le décodeur *maximum a posteriori* (MAP) qui cherche parmi toutes les séquences possibles $\mathbf{x}$, la séquence estimée $\hat{\mathbf{x}}$ pour laquelle la probabilité conditionnelle $Pr(\mathbf{x}|\mathbf{y})$ est la plus grande.

Dans ce paragraphe, nous allons nous intéresser au décodage à entrée et sortie pondérées. Ce décodeur calcule pour chaque bit, la probabilité conditionnelle *a posteriori* $APP(x_i = a)$ suivante :

$$\begin{aligned} APP(x_i = a) &= Pr\{\mathbf{x}_i = a|\mathbf{y}\} \\ &= \sum_{\mathbf{x}} Pr\{\mathbf{x}, x_i = a|\mathbf{y}\} \\ &= \sum_{\mathbf{x}:x_i=a} Pr\{\mathbf{x}|\mathbf{y}\} \\ &= \sum_{\mathbf{x}:x_i=a} \frac{p(\mathbf{y}|\mathbf{x})Pr\{\mathbf{x}\}}{p(\mathbf{y})} \\ &\propto \sum_{\mathbf{x}:x_i=a} p(\mathbf{y}|\mathbf{x})Pr\{\mathbf{x}\} \end{aligned} \tag{1.1}$$

Le signe $\propto$ signifie "proportionnel à".

$\sum_{\mathbf{x}:x_i=a}$ signifie que la somme est effectuée pour tous les mots de code $\mathbf{x}$ dont le bit en position i $x_i = a$.

Comme dans ce chapitre, nous nous restreindrons au cas de la modulation bipodale, a pourra prendre la valeur $+1$ ou -1.

Si on suppose que les probabilités conditionnelles $p(y_j|x_j)$ et les probabilités *a priori* $Pr\{x_j\}$ sont indépendantes, alors on peut écrire :

$$APP(x_i = a) \propto \sum_{\mathbf{x}:x_i=a} \prod_j p(y_j|x_j)Pr\{x_j\} \tag{1.2}$$

Finalement en sortant la probabilité *a priori* $Pr\{x_i = a\}$ et la probabilité conditionnelle $p(y_i|x_i)$ du signe somme, on obtient la relation classique suivante :

$$\boxed{APP(x_i = a) \propto Pr\{x_i = a\}p(y_i|x_i)EXTR(x_i = a)} \tag{1.3}$$

avec la probabilité extrinsèque $EXTR(x_i = a)$:

$$EXTR(x_i = a) = \sum_{\mathbf{x}:x_i=a} \prod_{j\neq i} p(y_j|x_j)Pr\{x_j\} \tag{1.4}$$

Ainsi la probabilité *a posteriori* $APP(x_i = a)$ est le produit de trois termes : la probabilité *a priori* $Pr\{x_i = a\}$, la probabilité conditionnelle $p(y_i|x_i)$ et la probabilité extrinsèque $EXTR(x_i = a)$.

Il est important de noter que le calcul de $EXTR(x_i = a)$ utilise toutes les informations disponibles $Pr\{x_j\}$ et $p(y_j|x_j)$ exceptées $Pr\{x_i = a\}$ et $p(y_i|x_i)$. Nous verrons que cette propriété est fondamentale pour le décodage itératif des codes concaténés.

Dans la relation (1.3), il est parfois plus pratique d'utiliser des logarithmes de rapport de vraisemblance LLR (*logarithm likelihood ratio* en anglais) à la place des probabilités lorsque a peut prendre les valeurs $+1$ ou -1.

La relation (1.3) s'écrit alors :

$$\ln \frac{APP(x_i = +1)}{APP(x_i = -1)} = \ln \frac{Pr(x_i = +1)}{Pr(x_i = -1)} + \ln \frac{p(y_i|x_i = +1)}{p(y_i|x_i = -1)} + \ln \frac{EXTR(x_i = +1)}{EXTR(x_i = -1)} \tag{1.5}$$

Cette relation s'écrit simplement

$$L_{APP}(x_i) = L_{APRI}(x_i) + L_{INT}(x_i) + L_{EXTR}(x_i) \tag{1.6}$$

avec

$$L_{APP}(x_i) = \ln \frac{APP(x_i = +1)}{APP(x_i = -1)}$$

$$L_{APRI}(x_i) = \ln \frac{Pr(x_i = +1)}{Pr(x_i = -1)}$$

$$L_{INT}(x_i) = \ln \frac{p(y_i|x_i = +1)}{p(y_i|x_i = -1)}$$

$$L_{EXTR}(x_i) = \ln \frac{EXTR(x_i = +1)}{EXTR(x_i = -1)}$$

$L_{APRI}(x_i)$, $L_{INT}(x_i)$, $L_{EXTR}(x_i)$ et $L_{APP}(x_i)$ sont respectivement les logarithmes du rapport de vraisemblance *a priori*, intrinsèque ou issu du canal de transmission, extrinsèque et *a posteriori*.

Finalement on peut aussi exprimer $p(y_i|x_i = +1)$ et $p(y_i|x_i = -1)$ en fonction de $L_{INT}(x_i)$. A partir de la définition de $L_{INT}(x_i)$, on a :

$$p(y_i|x_i = +1) = \exp(L_{INT}(x_i))p(y_i|x_i = -1) \tag{1.7}$$

En multipliant et divisant par $1 + \exp(L_{INT}(x_i))$, cette relation s'écrit :

$$p(y_i|x_i = +1) = \frac{\exp(L_{INT}(x_i))}{1 + \exp(L_{INT}(x_i))}(p(y_i|x_i = -1) + p(y_i|x_i = -1)) \tag{1.8}$$

Par ailleurs, on a :

$$p(y_i|x_i = -1) = \frac{1}{1 + \exp(L_{INT}(x_i))}(p(y_i|x_i = -1) + p(y_i|x_i = -1)) \tag{1.9}$$

Finalement on a les relations suivantes :

$$p(y_i|x_i = +1) \propto \frac{\exp(L_{INT}(x_i))}{1 + \exp(L_{INT}(x_i))} \tag{1.10}$$

$$p(y_i|x_i = -1) \propto \frac{1}{1 + \exp(L_{INT}(x_i))} \tag{1.11}$$

où $\propto$ signifie proportionnel à.

1.2.2 Application au canal à bruit blanc additif gaussien

On considère une séquence $\mathbf{x}$ codée par un code de rendement R. Cette séquence est transmise sur un canal additif à bruit blanc gaussien en utilisant une modulation bipodale. L'échantillon reçu en sortie du filtre adapté y_i est égal à :

$$y_i = \sqrt{RE_b}x_i + n_i \tag{1.12}$$

E_b est l'énergie par bit utile et n_i est l'échantillon de bruit de variance $\sigma^2 = \frac{N_0}{2}$
Comme le bruit est gaussien, la probabilité conditionnelle $p(y_i|x_i)$ s'écrit :

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y_i - x_i)^2}{2\sigma^2}\right) \tag{1.13}$$

Calculons l'information intrinsèque $L_{INT}(x_i)$:

$$\begin{aligned} L_{INT}(x_i) &= \ln \frac{p(y_i|x_i=+1)}{p(y_i|x_i=-1)} \\ &= \ln \frac{\frac{1}{\sqrt{2\sigma^2}} \exp\left(\frac{-(y_i-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\sigma^2}} \exp\left(\frac{-(y_i+1)^2}{2\sigma^2}\right)} \\ &= \frac{2y_i}{\sigma^2} \end{aligned} \tag{1.14}$$

Ainsi pour le canal BBAG, le signal reçu y_i est proportionnel à l'information intrinsèque $L_{INT}(x_i)$. Cette propriété permet de comprendre pourquoi on utilise généralement les logarithmes de rapport de vraisemblance pour le décodage à entrées et sorties pondérées.

1.2.3 Cas du code de parité (3,2)

Le graphe factorisé [1] associé à ce code est présenté sur la figure 1.1.

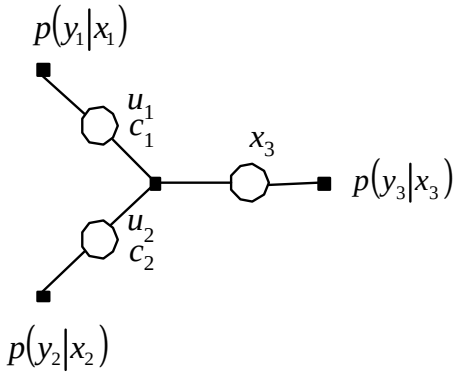


FIG. 1.1 – Graphe factorisé du code de parité (3,2).

A partir des échantillons reçus y_1, y_2 et y_3 on peut calculer les probabilités conditionnelles $p(y_1|x_1=-1)$, $p(y_1|x_1=1)$, $p(y_2|x_2=-1)$, $p(y_2|x_2=1)$, $p(y_3|x_3=-1)$ et $p(y_3|x_3=1)$. En appliquant la relation (1.3) on obtient pour $APP(x_1=a)$

$$\begin{aligned} APP(x_1=a) &= Pr\{x_1=a|y_1,y_2,y_3\} \\ &\propto Pr\{y_1|x_1=a\}Pr\{x_1=a\}EXTR(x_1=a) \end{aligned} \tag{1.15}$$

avec

$$\begin{aligned} EXTR(x_1=+1) &= \sum_{\mathbf{x}:x_i=1} \prod_{j\neq 1} p(y_j|x_j)Pr\{x_j\} \\ &= p(y_2|x_2=-1)Pr\{x_2=-1\}p(y_3|x_3=+1)Pr\{x_3=+1\} \\ &\quad + p(y_2|x_2=+1)Pr\{x_2=+1\}p(y_3|x_3=-1)Pr\{x_3=-1\} \end{aligned} \tag{1.16}$$

[1] Par rapport aux graphes de Tanner et aux graphes TWL, on a ajouté sur les graphes factorisés des noeuds spécifiques correspondant aux messages reçus à la sortie du canal de transmission. Ces graphes sont appelés graphes factorisés car ils permettent de factoriser la probabilité conjointe a posteriori $Pr\{\mathbf{x};\mathbf{y}\}$ pour une séquence en entrée du canal $\mathbf{x}$ et l'observation $\mathbf{y}$.

$$EXTR(x_1=-1) = \sum_{\mathbf{x}:x_i=-1} \prod_{j\neq 1} p(y_j|x_j)Pr\{x_j\}$$
$$= p(y_2|x_2=-1)Pr\{x_2=-1\}p(y_3|x_3=-1)Pr\{x_3=-1\}$$
$$+ p(y_2|x_2=+1)Pr\{x_2=+1\}p(y_3|x_3=+1)Pr\{x_3=+1\} \quad (1.17)$$

1.2.4 Cas du code de répétition (3,1)

Le graphe factorisé associé à ce code est présenté sur la figure 1.2.

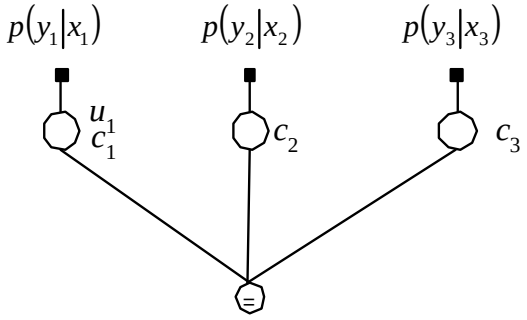


FIG. 1.2 – Graphe factorisé du code de répétition (3,1).

A partir des échantillons reçus y_1, y_2 et y_3 on peut calculer les probabilités conditionnelles $p(y_1|x_1=-1)$, $p(y_1|x_1=+1)$, $p(y_2|x_2=-1)$, $p(y_2|x_2=+1)$, $p(y_3|x_3=-1)$, $p(y_3|x_3=+1)$. En appliquant la relation (1.3) on obtient pour $APP(x_1=a)$

$$APP(x_1=a) = Pr\{x_1=a|y_1,y_2,y_3\}$$
$$\propto Pr\{y_1|x_1=a\}Pr\{x_1=a\}EXTR(x_1=a) \quad (1.18)$$

avec

$$EXTR(x_1=+1) = \sum_{\mathbf{x}:x_i=1} \prod_{j\neq 1} p(y_j|x_j)Pr\{x_j\}$$
$$= p(y_2|x_2=+1)Pr\{x_2=+1\}p(y_3|x_3=+1)Pr\{x_3=+1\} \quad (1.19)$$

$$EXTR(x_1=-1) = \sum_{\mathbf{x}:x_i=-1} \prod_{j\neq 1} p(y_j|x_j)Pr\{x_j\}$$
$$= p(y_2|x_2=-1)Pr\{x_2=-1\}p(y_3|x_3=-1)Pr\{x_3=-1\} \quad (1.20)$$

1.3 Algorithme Somme-Produit

1.3.1 Introduction

L'algorithme Somme-Produit permet un calcul efficace de toutes les probabilités *a posteriori* $APP(x_i=a)$. Différents algorithmes issus du traitement du signal et des communications numériques peuvent être décrits avec ce simple algorithme. Cet algorithme est aussi connu sous le nom d'algorithme de propagation de croyance BP (*belief propagation* en anglais) dans le domaine de l'intelligence artificielle. L'algorithme Somme-Produit est appliqué sur le graphe factorisé du code. Les messages qui transitent sur les branches sont des probabilités ou des logarithmes de rapport de vraisemblance.

1.3.2 Règles de base

A partir des résultats obtenus pour le code de parité (3,2) et le code de répétition (3,1) nous allons déterminer les règles de décodage lorsque l'on applique l'algorithme Somme-Produit sur un graphe dont les nœuds de variable sont binaires et dont les nœuds de contrôle sont des équations de parité (additions modulo 2).

Nous allons exprimer la première règle de calcul de l'information extrinsèque. Soit le nœud de variable binaire est de degré 3 comme sur la figure 1.3 (a).

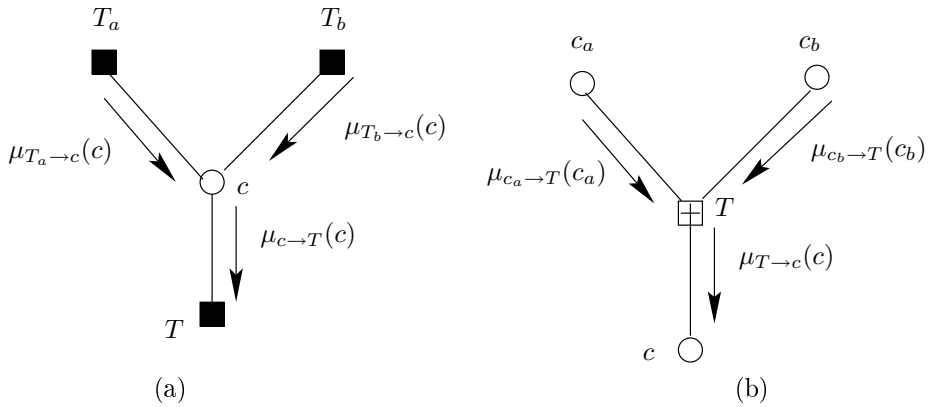


FIG. 1.3 – Flot de message pour les calculs de $\mu_{c \to T}(c)$ et $\mu_{T \to c}(c)$.

Soit $\mu_{T_i \to c}(c)$ le message du nœud de contrôle T_i vers le nœud de variable binaire c. Ce message est la densité de probabilité de la variable binaire c conditionnée par les informations issues du nœud T_a. Le message $\mu_{T_i \to c}(c)$ est un vecteur (p_{i0}, p_{i1}) avec $p_{i0} + p_{i1} = 1$. A partir du calcul de l'information extrinsèque du code de répétition (3,1) on peut calculer le message $\mu_{c \to T}(c)$. La première règle est la suivante :

$$\mu_{c \to T}(c) = \left(\frac{p_{a0}p_{b0}}{p_{a1}p_{b1} + p_{a0}p_{b0}}, \frac{p_{a1}p_{b1}}{p_{a1}p_{b1} + p_{a0}p_{b0}}\right) \tag{1.21}$$

Le terme $p_{a1}p_{b1} + p_{a0}p_{b0}$ au dénominateur est un terme de normalisation.

On considère un nœud de contrôle T de degré 3 comme sur la figure 1.3 (b). Soit $\mu_{c_i \to T}(c_i) = (p_{i0}, p_{i1})$ le message du nœud de variable c_i vers le nœud de contrôle T. A partir du calcul de l'information extrinsèque du code de parité (3,1) on peut calculer $\mu_{T \to c}(c)$. On obtient la seconde règle de calcul suivante :

$$\mu_{T \to c}(c) = (p_{a0}p_{b0} + p_{a1}p_{b1}, p_{a1}p_{b0} + p_{a0}p_{b1}) \tag{1.22}$$

On peut aussi utiliser les logarithmes du rapport de vraisemblance pour décrire les messages $\mu_{c \to T}(c)$ et $\mu_{T \to c}(c)$. Soit

$$\Lambda_i = \ln\left(\frac{p_{i1}}{p_{i0}}\right) \tag{1.23}$$

Nous avons les relations suivantes entre Λ_i et p_{i0} et p_{i1} :

$$p_{i1} = \frac{\exp(\Lambda_i)}{1 + \exp(\Lambda_i)} \quad \text{et} \quad p_{i0} = \frac{1}{1 + \exp(\Lambda_i)} \tag{1.24}$$

A partir de la première règle, on obtient :

$$\mu_{c \to T}(c) = \mu_{T_a \to c}(c) + \mu_{T_b \to c}(c) \tag{1.25}$$

La seconde règle s'écrit :

$$\begin{aligned}\mu_{T\to c}(c) &= \ln \frac{\exp(\Lambda_a)+\exp(\Lambda_b)}{\exp(\Lambda_a)\exp(\Lambda_b)+1} \\ &= -2\tanh^{-1}\left(\tanh\left(\frac{\mu_{c_a\to T}(c_a)}{2}\right)\tanh\left(\frac{\mu_{c_b\to T}(c_b)}{2}\right)\right)\end{aligned} \tag{1.26}$$

en utilisant la relation $\tanh(u/2) = (\exp(u)-1)/(\exp(u)+1)$ [2][22].

Lorsque les nœuds sont de degré supérieur à 3, les règles deviennent :

$$\mu_{c\to T}(c) = \sum_{n(c)\backslash T} \mu_{T_i\to c}(c) \tag{1.27}$$

où $n(c)$ est l'ensemble des nœuds de contrôle reliés au nœud de variable c et $n(c)\backslash T$ signifie l'ensemble $n(c)$ excepté le nœud de contrôle T.

$$\mu_{T\to c}(c) = -2\tanh^{-1}\left(\prod_{n(T)\backslash c}\tanh\left(\frac{\mu_{c_i\to T}(c_i)}{2}\right)\right) \tag{1.28}$$

où $n(T)$ est l'ensemble des nœuds de variable reliés au nœud de parité T et $n(T)\backslash c$ signifie l'ensemble $n(T)$ excepté le nœud de variable c.

L'algorithme Minimum-Somme s'obtient à partir de l'algorithme Somme-Produit en remplaçant l'équation (1.26) par l'approximation suivante :

$$\begin{aligned}\mu_{T\to c}(c) &= -\min\left(|\mu_{c_a\to T}(c_a)|, |\mu_{c_b\to T}(c_b|\right)\mathrm{sgn}(\mu_{c_a\to T}(c_a))\mathrm{sgn}(\mu_{c_b\to T}(c_b)) \\ &= -\mu_{c_a\to T}(c_a)\boxplus\mu_{c_b\to T}(c_b)\end{aligned} \tag{1.29}$$

où

$$\mathrm{sgn(a)} = \begin{cases}+1 & \text{si} \quad a\geq 0 \\ -1 & \text{si} \quad a<0\end{cases} \tag{1.30}$$

et

$$a\boxplus b = \min(|a|,|b|)\mathrm{sgn}(a)\mathrm{sgn}(b) \tag{1.31}$$

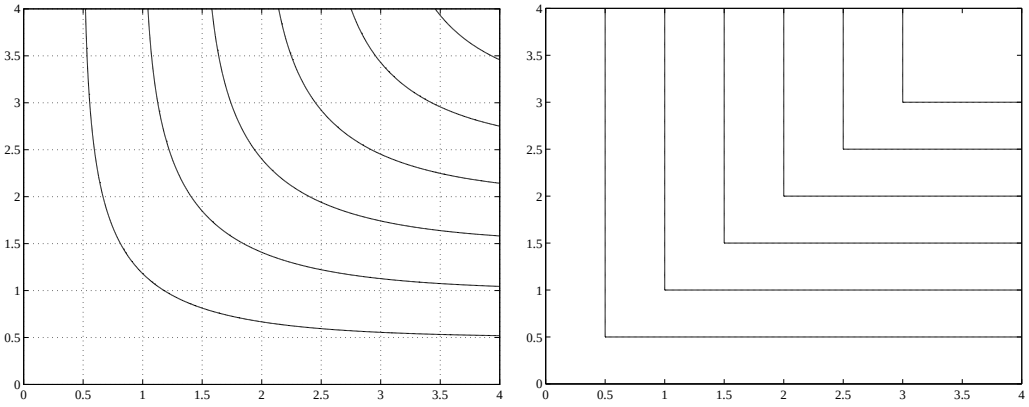
On utilisera l'opérateur $\boxplus$ pour simplifier l'écriture dans la suite du document.

Lorsque le nœud de contrôle est de degré supérieur à 3, la relation (1.28) devient en utilisant l'opérateur $\boxplus$:

$$\mu_{T\to c}(c) = -\sum_{n(T)\backslash c}\boxplus\quad \mu_{c_i\to T}(c_i) \tag{1.32}$$

Sur la figure 1.4 nous avons comparé les courbes de contour des messages $\mu_{T\to c}(c)$ calculés en utilisant l'algorithme Somme-Produit (a) et Minimum-Somme (b) dans le cas de nœuds de contrôle de degré 3. Ces courbes sont tracées pour $\mu_{T\to c}(c)$=0.5, 1, 1.5, 2, 2.5 et 3 .

Pour les fortes valeurs de $\mu_{c_i\to T}(c_i)$, les 2 algorithmes donnent les mêmes résultats.



(a) (b)

FIG. 1.4 – Courbes de contour de $\mu_{T\to c}(c)$ en fonction de $\mu_{c_a\to T}(c_a)$ et $\mu_{c_b\to T}(c_b)$ (a) algorithme Somme-Produit (b) algorithme Minimum-Somme.

1.3.3 Application de l'algorithme Somme-Produit sur les graphes factorisés sans cycle

L'application de l'algorithme Somme-Produit sur les graphes factorisés sans cycle consiste simplement à appliquer les règles décrites précédemment en commençant par les extrémités ou feuilles de l'arbre. L'algorithme se termine lorsque toutes les branches ont été explorées deux fois : une fois dans le sens aller et une fois dans le sens retour.

1.3.4 Exemple

Considérons le code linéaire en bloc (7,4) défini par la matrice de parité suivante :

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{1.33}$$

Les trois bits de parité sont obtenus comme suit :

$$\begin{aligned} u_1 + u_2 + c_5 = 0 \qquad & T_1 \\ u_1 + u_3 + c_6 = 0 \qquad & T_2 \\ u_1 + u_4 + c_7 = 0 \qquad & T_3 \end{aligned}$$

Le graphe de Tanner de ce code est donné sur la figure 1.5.

Par rapport au code de Hamming (7,4), la distance minimale de ce code n'est que de 2.

Soit la séquence $\mathbf{x} = [+1 - 1 - 1 + 1 + 1 + 1 - 1]$ associée au mot de code binaire $\mathbf{c} = [1001110]$. La séquence $\mathbf{x}$ est émise sur un canal additif à bruit blanc gaussien de variance $\sigma^2 = 0.75$.

Soit la séquence reçue $\mathbf{y} = [+1.2 - 0.4 - 1.5 - 0.4 + 1.2 - 0.75 - 1.5]$.

En utilisant directement les probabilités conditionnelles $p(y_i|x_i = -1)$ et $p(y_i|x_i = +1)$ calculées à partir des échantillons reçus y_i en utilisant les relations (1.24), nous pouvons appliquer l'algorithme Somme-Produit sur le graphe factorisé du code (7,4).

Les différentes étapes sont présentées sur la figure 1.6. On commence par propager les probabilités conditionnelles vers les nœuds de parité T_1, T_2 et T_3 (a). Puis en utilisant la seconde règle, on calcule les messages vers le nœud de variable c_1 (b). Ensuite, on propage les messages vers les nœuds de parité en utilisant la première règle (c). Finalement en utilisant la seconde règle, on obtient les probabilités extrinsèques $EXTR(x_i = a)$. associées aux nœuds de variable c_2, c_3, c_4, c_5, c_6 et c_7 (d).

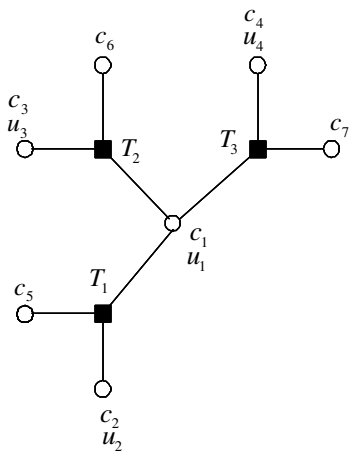


FIG. 1.5 – Graphe de Tanner du code$(7,4)$.

Par exemple pour le bit c_2 on a $EXTR(x_2 = +1) = 0.46$ et $EXTR(x_2 = -1) = 0.54$. Les probabilités *a posteriori* se calculent alors en utilisant la relation (1.3) :

$$\begin{aligned} APP(x_2 = +1) &\propto Pr\{x_2 = +1\}p(y_2|x_2 = +1)EXTR(x_2 = +1) \\ &\propto 0.5 \times 0.27 \times 0.46 = 0.0621 \end{aligned} \tag{1.34}$$

$$\begin{aligned} APP(x_2 = -1) &\propto Pr\{x_2 = -1\}p(y_2|x_2 = -1)EXTR(x_2 = -1) \\ &\propto 0.5 \times 0.73 \times 0.54 = 0.1971 \end{aligned} \tag{1.35}$$

En normalisant on obtient finalement

$$APP(x_2 = +1) = 0.2396$$

$$APP(x_2 = -1) = 0.7604$$

En utilisant directement les LLR intrinsèques $L_{INT}(x_i)$ ($L_{INT}(x_i) = \frac{2y_i}{\sigma^2}$), nous pouvons également appliquer l'algorithme Somme-Produit ou l'algorithme Minimum-Somme sur le graphe factorisé du code (7,4). Les différentes étapes de l'algorithme Minimum-Somme sont présentées sur la figure 1.7.

On a :

$$\{L_{INT}(x_i)\} = [+3 - 1 - 4 - 1 + 3 - 2 - 4]$$

En appliquant l'algorithme Minimum-Somme, on obtient les LLR extrinsèques $L_{EXTR}(x_i)$ suivants :

$$\{L_{EXTR}(x_i)\} = [-2 \quad 0 + 2 + 2 \quad 0 + 3 + 1]$$

Les LLR *a posteriori* $L_{APP}(x_i)$ peuvent alors être calculés en utilisant la relation $L_{APP}(x_i) = L_{APRI}(x_i) + L_{INT}(x_i) + L_{EXTR}(x_i)$. Finalement on obtient :

$$\{L_{APP}(x_i)\} = [+1 - 1 - 2 + 1 + 3 + 1 - 3]$$

car $L_{APRI}(x_i) = 0$ en absence d'information *a priori*.

Bien que l'algorithme Minimum-Somme ne soit pas un algorithme de décodage (l'objectif principal est de délivrer une information de fiabilité sur les bits décodés), si on réalise un seuillage des LLR *a posteriori*, on retrouve bien le mot de code émis :

$$\hat{\mathbf{x}} = [+1 - 1 - 1 + 1 + 1 + 1 - 1]$$

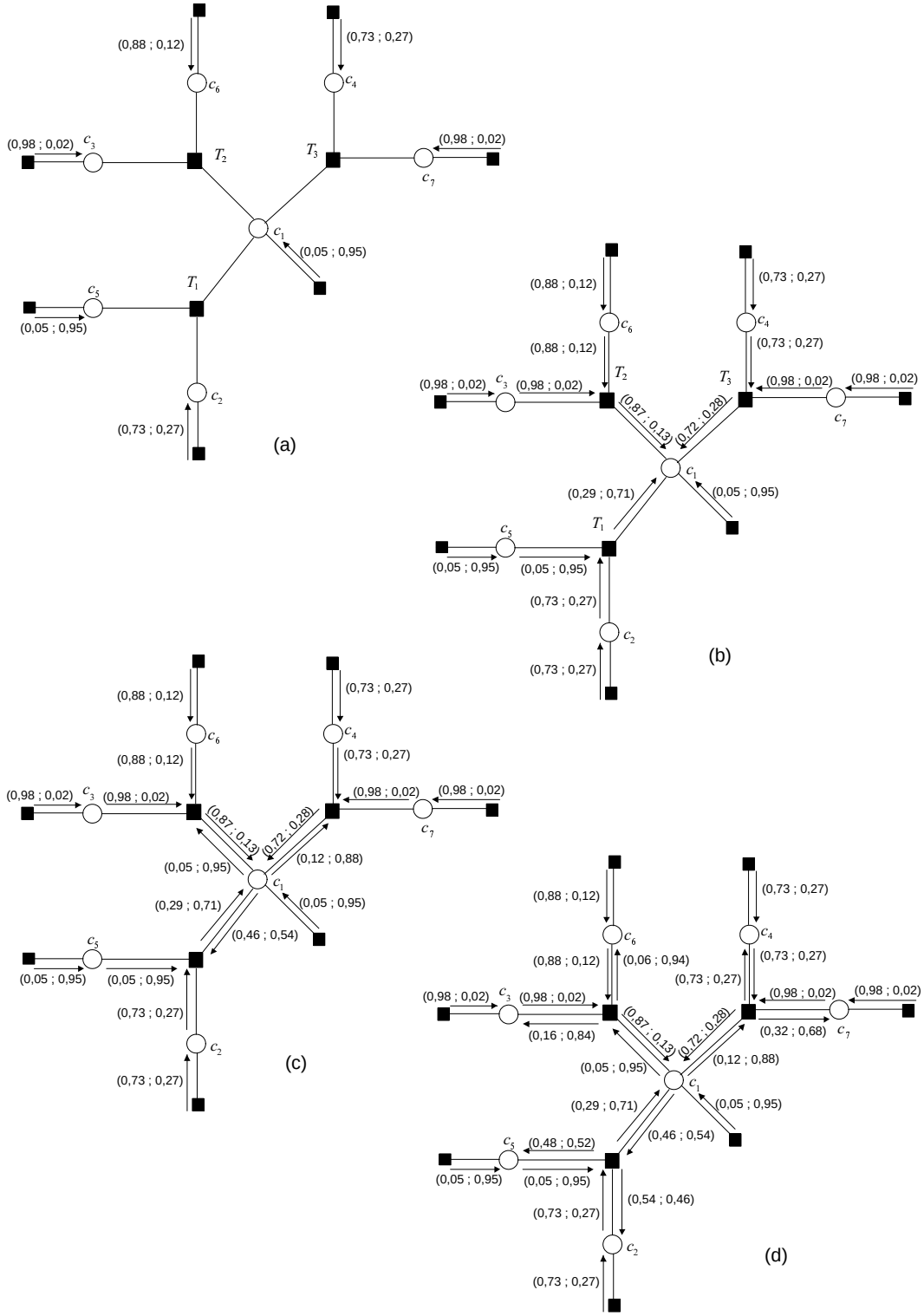


FIG. 1.6 – Algorithme Somme-Produit sur code$(7, 4)$.

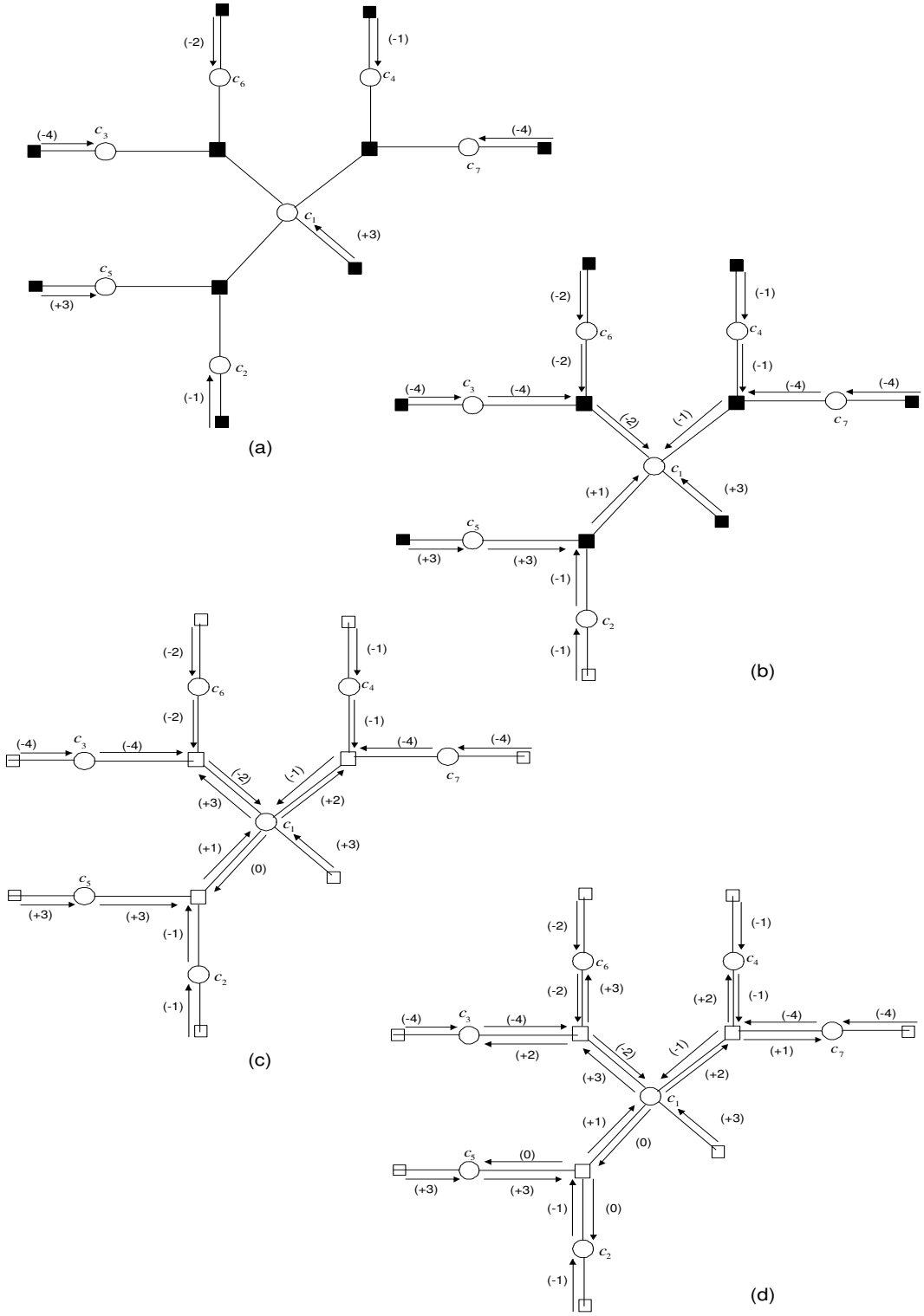


FIG. 1.7 – Algorithme Minimum-Somme sur code$(7, 4)$.

1.4 Algorithme Aller-Retour

L'algorithme Aller-Retour ou BCJR [1] estime les probabilités *a posteriori* des états et des transitions d'une source de Markov observée à travers un canal discret, bruité et sans mémoire.

Un codeur convolutif binaire de rendement k/n peut être vu comme une source markovienne de mémoire 1 et avec $S = 2^M$ états internes, où M est le nombre de cellules mémoires du codeur.

On considère un codeur convolutif binaire de rendement k/n avec $S = 2^M$ états internes, où M est le nombre de cellules mémoires du codeur convolutif.

Soit $\mathbf{u}_t = (u_t^1, u_t^2, \ldots, u_t^k)$ et $\mathbf{c}_t = (c_t^1, c_t^2, \ldots, c_t^n)$ respectivement l'entrée et la sortie du codeur convolutif à l'instant t. Soit $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$ la séquence associée émise sur le canal de transmission.

En sortie du canal discret bruité et sans mémoire le décodeur reçoit $y_t = (y_t^1, y_t^2, \ldots, y_t^n)$ à l'instant t . Dans ce paragraphe, nous nous limiterons au cas des codes convolutifs invariants dans le temps.

L'algorithme Aller-Retour ou BCJR [1] permet d'estimer les probabilités *a posteriori* conjointes $\sigma_t(m', m)$ d'un codeur convolutif observé à travers un canal discret, bruité et sans mémoire :

$$\sigma_t(m', m) = Pr\{s_t = m'; s_{t+1} = m; \mathbf{y}_0^{T-1}\} \tag{1.36}$$

où s_t est l'état à l'instant t avec $s_t = m, m \in \{0, 1, \ldots, S-1\}$ et $\mathbf{y}_0^{T-1}$ est la séquence reçue de l'instant 0 à $T-1$.

A partir de ces probabilités, on peut déterminer la probabilité *a posteriori* (APP) sur x_t^i. On a :

$$\begin{aligned} APP(x_t^i = a) &= Pr\{x_t^i = a|\mathbf{y}_0^{T-1}\} \\ &\propto \sum_{m', m/x_t^i = a} \sigma_t(m', m) \end{aligned} \tag{1.37}$$

On calcule $\sigma_t(m', m)$ en utilisant la relation suivante :

$$\sigma_t(m', m) = \alpha_t(m')\gamma_t(m', m)\beta_{t+1}(m) \tag{1.38}$$

Avec

$$\alpha_t(m') = Pr\{s_t = m'; \mathbf{y}_0^{t-1}\} \tag{1.39}$$

$$\gamma_t(m', m) = Pr\{s_{t+1} = m; \mathbf{y}_t|s_t = m'\} \tag{1.40}$$

$$\beta_t(m) = Pr\{\mathbf{y}_t^{T-1}|s_t = m\} \tag{1.41}$$

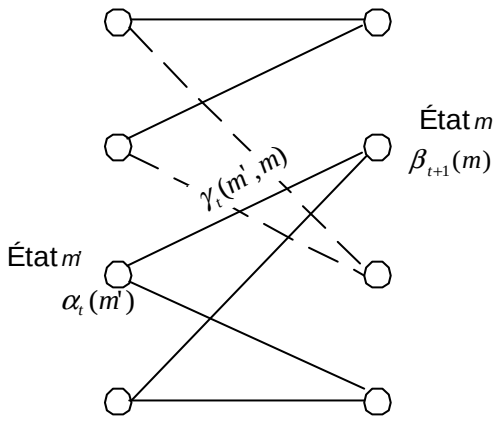
Les quantités $\alpha_t(m)$, $\gamma_t(m', m)$ et $\beta_{t+1}(m)$ représentent l'influence de la séquence reçue respectivement avant l'instant t, à l'instant t et après l'instant t sur la probabilité conjointe $\sigma_t(m', m)$.

Le calcul de la métrique de branche $\gamma_t(m', m)$ s'exprime comme suit :

$$\begin{aligned} \gamma_t(m', m) &= Pr\{s_{t+1} = m; \mathbf{y}_t|s_t = m'\} \\ &= Pr\{s_{t+1} = m|s_t = m'\} \times Pr\{\mathbf{y}_t|s_{t+1} = m; s_t = m'\} \\ &= Pr\{s_{t+1} = m|s_t = m'\} \sum_{\mathbf{x}_t \in \mathbb{X}} Pr\{\mathbf{y}_t, \mathbf{x}_t|s_t = m'; s_{t+1} = m\} \\ &= Pr\{s_{t+1} = m|s_t = m'\} \sum_{\mathbf{x}_t \in \mathbb{X}} p(\mathbf{y}_t|\mathbf{x}_t) Pr\{\mathbf{x}_t = \mathbf{x}|s_t = m'; s_{t+1} = m\} \end{aligned} \tag{1.42}$$

où la probabilité *a posteriori* sur la séquence de sortie conditionnellement à la séquence d'entrée est égale à :

$$Pr\{\mathbf{y}_t|\mathbf{x}_t\} = \prod_{j=1}^{n} p(y_t^j/x_t^j) \tag{1.43}$$



et $p(y_t^j/x_t^j)$ est la probabilité de transition du canal.

La probabilité $Pr\{\mathbf{x}_t = \mathbf{x}|s_t = m', s_{t+1} = m\}$ est égale à 0 ou 1 selon $\mathbf{x}_t(m', m)$:

$$Pr\{\mathbf{x}_t = \mathbf{x}|s_t = m', s_{t+1} = m\} = \begin{cases} 1 & \text{si} \quad \mathbf{x}_t(m', m) = \mathbf{x} \\ 0 & \text{sinon} \end{cases} \tag{1.44}$$

Le calcul des probabilités $\alpha_t(m)$ est effectué par récurrence :

$$\alpha_t(m) = \sum_{m'=0}^{S-1} \alpha_{t-1}(m')\gamma_{t-1}(m', m) \tag{1.45}$$

De même que le calcul des probabilités $\beta_t(m)$:

$$\beta_t(m) = \sum_{m'=0}^{S-1} \beta_{t+1}(m')\gamma_t(m', m) \tag{1.46}$$

En utilisant (1.38) et (1.42) on obtient :

$$APP(x_t^i = a) \propto \sum_{m', m|x_t^i=a} \alpha_t(m')Pr\{s_{t+1} = m|s_t = m'\}\Big(\sum_{\mathbf{x}_t \in \mathbb{X}} p(\mathbf{y}_t|\mathbf{x}_t)Pr\{\mathbf{x}_t = \mathbf{x}|s_t = m'; s_{t+1} = m\}\Big)\beta_{t+1}(m) \tag{1.47}$$

Finalement, en utilisant En utilisant (1.43) et (1.44) l'expression (1.47) peut s'écrire :

$$APP(x_t^i = a) \propto \sum_{m', m/x_t^i=a} \alpha_t(m')Pr\{s_{t+1} = m|s_t = m'\}\prod_{j=1}^{n} p(y_t^j/x_t^j(m', m))\beta_{t+1}(m) \tag{1.48}$$

L'expression finale des probabilités *a posteriori* $APP(x_t^i = a)$ et extrinsèque $EXTR(x_t^i = a)$ dépend de la structure du codeur convolutif.

Nous pouvons décrire l'algorithme Aller-Retour en 4 étapes :

- **Initialisation**
  L'état initial et l'état final du codeur sont égaux à l'état zéro

$$\alpha_0(0) = 1 \qquad \text{et} \qquad \alpha_0(m) = 0 \quad \forall m \neq 0$$

$$\beta_T(0) = 1 \qquad \text{et} \qquad \beta_T(m) = 0 \quad \forall m \neq 0$$

- **Calcul des γs et Procédure aller**
  Les γs sont calculés pour toutes les branches en utilisant la relation (1.42) et simultanément les probabilités $\alpha_t(m)$ sont calculées à partir de (1.45)
- **Procédure retour**
  Lorsque la séquence $\mathbf{y}_0^{T-1}$ est reçue, les probabilités $\beta_t(m)$ sont calculées en utilisant la relation (1.46)
- **Calcul des probabilités *a posteriori***
  Finalement les probabilités *a posteriori* sont calculées en utilisant la relation (1.38).

Codes convolutifs non systématiques

En introduisant $APRI(x_t^i)$ la probabilité *a priori* sur x_t^i , $Pr\{s_{t+1} = m | s_t = m'\}$ peut s'écrire :

$$Pr\{s_{t+1} = m | s_t = m'\} = \prod_{i=1}^{n} APRI(x_t^i = x_t^i(m', m)) \tag{1.49}$$

où $x^i(m', m)$ est la valeur de la sortie associée à la branche (m', m)

Si aucune probabilité *a priori* n'est disponible à l'entrée du décodeur, alors $APRI(x_t^i) = 1/2$ et $p(m|m')$ est égal à 2^n .

On a :

$$\begin{aligned} APP(x_t^i = a) &= \sum_{m', m/x_t^i = a} \alpha_t(m') \prod_{i'=1}^{n} APRI(x_t^{i'} = x_t^{i'}(m', m)) \prod_{j=1}^{n} p(y_t^j \mid x_t^j(m', m)) \beta_{t+1}(m) \\ &= APRI(x_t^i = a) \times EXTR(x_t^i = a) \end{aligned} \tag{1.50}$$

avec

$$EXTR(x_t^i = a) = \sum_{m', m/x_t^i = a} \alpha_t(m') \prod_{\substack{i'=1 \\ i' \neq i}}^{n} APRI(x_t^{i'} = x_t^{i'}(m', m)) \prod_{j=1}^{n} p(y_t^j \mid x_t^j(m', m)) \beta_{t+1}(m) \tag{1.51}$$

1.4.1 Codes convolutifs systématiques

$$\begin{aligned} APP(x_t^i = a) &\propto \sum_{m', m/x_t^i = a} \alpha_t(m') \prod_{i'=1}^{n} APRI(x_t^{i'} = x_t^{i'}(m', m)) \\ &\quad \times \prod_{i'=1}^{n} p(y_t^{i'} \mid x_t^{i'}(m', m)) \beta_{t+1}(m) \\ &\propto APRI(x_t^i = a) \times p(y_t^i \mid x_t^i = a) \times EXTR(x_t^i = a) \end{aligned} \tag{1.52}$$

avec

$$EXTR(x_t^i = a) = \sum_{m',m/x_t^i=a} \alpha_t(m') \prod_{\substack{i'=1 \\ i' \neq i}}^{n} APRI(x_t^{i'} = x^{i'}(m',m)) \times \prod_{\substack{i'=1 \\ i' \neq i}}^{n} p(y_t^{i'} \mid x_t^{i'}(m',m)) \beta_{t+1}(m) \tag{1.53}$$

Codes convolutifs systématiques de rendement 1/2

Soit $\mathbf{x}_t = (x_t^S, x_t^P)$ le mot émis sur le canal à l'instant t et $\mathbf{y}_t = (y_t^S, y_t^P)$ les échantillons reçus à la sortie du canal sans mémoire. On a alors :

$$\begin{aligned} APP(x_t^S = a) &\propto \sum_{m',m/x_t^S=a} \alpha_t(m') \times APRI(x_t^S = x_t^S(m',m)) \times APRI(x_t^P = x_t^P(m',m)) \\ &\quad \times p(y_t^S \mid x_t^S(m',m)) \times p(y_t^P \mid x_t^P(m',m)) \times \beta_{t+1}(m) \\ &\propto APRI(x_t^S = a) \times p(y_t^S \mid x_t^S = a) \times EXTR(x_t^S = a) \end{aligned} \tag{1.54}$$

avec

$$EXTR(x_t^S = a) = \sum_{m',m/x_t^S=a} \alpha_t(m') \times APRI(x_t^P = x_t^P(m',m)) \times p(y_t^P \mid x_t^P(m',m)) \times \beta_{t+1}(m) \tag{1.55}$$

En général, nous n'avons aucune information *a priori* sur x_t^P ; l'information extrinsèque $EXTR(x_t^S = a)$ devient alors :

$$EXTR(x_t^S = a) = \sum_{m',m/x_t^S=a} \alpha_t(m') \times p(y_t^P \mid x_t^P(m',m)) \times \beta_{t+1}(m) \tag{1.56}$$

1.4.2 Exemple

Considérons un codeur convolutif récursif systématique de rendement $1/2$ $G(D) = \left(1 \quad \frac{1}{1+D}\right)$. Le treillis associé est présenté sur la figure 1.8.

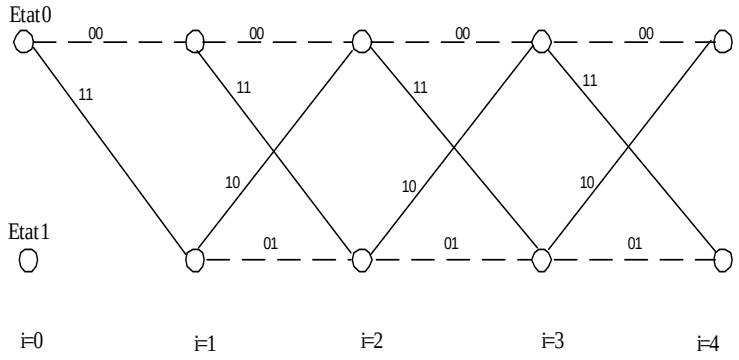


FIG. 1.8 – treillis du codeur $\left(1 \quad \frac{1}{1+D}\right)$

Soit le mot d'information et le mot de code associé

i	0	1	2	3	4	5
u_i	1	0	0	1	1	1
x_i^S	+1	-1	-1	+1	+1	+1
x_i^P	+1	+1	+1	-1	+1	-1

Soient les informations intrinsèques L_{INT} reçues du canal BBAG (obtenues après normalisation) :

$L_{INT}(x_i^S)$	+1.21	-1.00	-2.86	+4.84	+0.90	+5.87
$L_{INT}(x_i^P)$	+1.09	-1.07	+3.49	+0.72	+0.77	-1.13

1) calcul des métriques de branche γ

Pour un code convolutif systématique de rendement $1/2$, le calcul de la métrique de branche est le suivant :

$$\gamma_t(m', m) = APRI(x_t^S = x_t^S(m', m)) \times p(y_t^S \mid x_t^S(m', m)) \times p(y_t^P \mid x_t^P(m', m)) \tag{1.57}$$

Calculons $p(y_t \mid x_t)$ pour un canal BBAG avec $x_t = \pm 1$:

$$
\begin{aligned}
p(y_t \mid x_t = a) &= \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(\frac{-(y_t - x_t)^2}{2\sigma^2}\right) \\
&\propto \exp\left(\frac{2y_t x_t}{2\sigma^2}\right) \\
&\propto \exp\left(\frac{x_t}{2} L_{INT}(x_t)\right) \\
&\propto \exp\left(\frac{(x_t + 1)}{2} L_{INT}(x_t)\right) \qquad (1.58)
\end{aligned}
$$

Le passage de la seconde à la troisième ligne utilise la relation $L_{INT}(x_t) = \frac{2}{\sigma^2} y_t$. La dernière ligne permet de simplifier les calculs.

Dans notre exemple, en absence d'information *a priori*, les $2^n = 4$ métriques de branche γ_t^{pq} où p et q sont respectivement les bits relatifs à x_t^S et x_t^P sont données dans la table suivante :

γ_i^{00}	1	1	1	1	1	1
γ_i^{10}	3.35	0.36	0.05	126.46	2.45	354.24
γ_i^{01}	2.97	0.34	32.78	2.05	2.16	0.32
γ_i^{11}	9.97	0.12	1.87	259.81	5.31	114.42

2) calcul des α

$i = 0$: initialisation

$\alpha_0(0) = 1$
$\alpha_0(1) = 0$

$i = 1$

$\alpha_1(0) \propto \alpha_0(0)\gamma_0^{00} = 1$
$\alpha_1(1) \propto \alpha_0(0)\gamma_0^{11} = 9.97$
$\alpha_1(0) = 0.09$
$\alpha_1(1) = 0.91$

$i = 2$

$\alpha_2(0) \propto \alpha_1(0)\gamma_1^{00} + \alpha_1(1)\gamma_1^{10} = 0.417$
$\alpha_2(1) \propto \alpha_1(0)\gamma_1^{11} + \alpha_1(1)\gamma_1^{01} = 0.32$
$\alpha_2(0) = 0.57$
$\alpha_2(1) = 0.43$
$\vdots$

Les résultats finaux sont présentés sur la figure 1.9

3) calcul des β

$i = 6$: initialisation

$\beta_6(0) = 0.5$
$\beta_6(1) = 0.5$ car le trellis n'a pas été terminé.

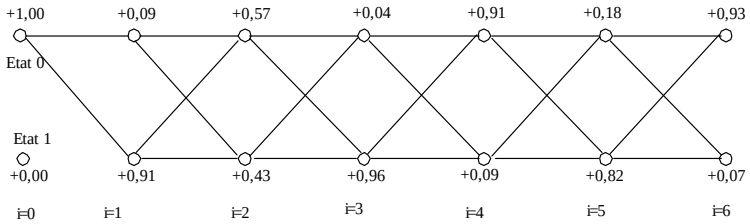


FIG. 1.9 – calcul des α

$i = 5$

$\beta_5(0) \propto \beta_6(0)\gamma_5^{00} + \beta_6(1)\gamma_5^{11} = 57.71$
$\beta_5(1) \propto \beta_6(0)\gamma_5^{10} + \beta_6(1)\gamma_5^{01} = 177.28$
$\beta_5(0) = 0.25$
$\beta_5(1) = 0.75$
$\vdots$

Les résultats finaux sont présentés sur la figure 1.10

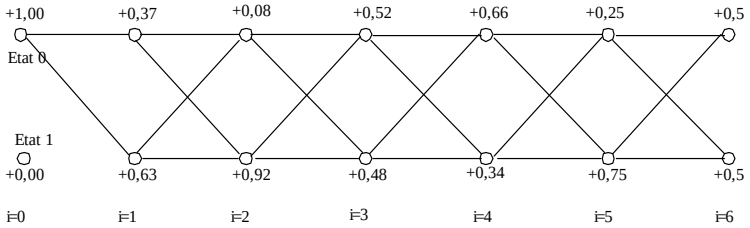


FIG. 1.10 – calcul des β

4) calcul de $L_{APP}(x_i^S)$

On utilise la relation

$$APP(x_i^S = a) \propto \sum_{m',m/x_i^S=a} \alpha_t(m') \times \gamma_t(m',m) \times \beta_{t+1}(m) \tag{1.59}$$

$i = 0$

$APP(x_0^S = -1) \propto \alpha_0(0) \times \gamma_0^{00} \times \beta_1(0) = 0.37$
$APP(x_0^S = +1) \propto \alpha_0(0) \times \gamma_0^{11} \times \beta_1(1) = 6.28$
$L_{APP}(x_0^S) = +2.85$

$i = 1$

$APP(x_1^S = -1) \propto \alpha_1(0) \times \gamma_1^{00} \times \beta_2(0) + \alpha_1(1) \times \gamma_1^{01} \times \beta_2(1) = 0.2918$
$APP(x_1^S = +1) \propto \alpha_1(0) \times \gamma_1^{11} \times \beta_2(1) + \alpha_1(1) \times \gamma_1^{10} \times \beta_2(0) = 0.0361$
$L_{APP}(x_1^S) = -2.04$

$\vdots$

Finalement après calcul, on obtient les informations *a posteriori* suivantes :

i	0	1	2	3	4	5
$L_{APP}(x_i^S)$	+2.85	-2.04	-2.61	+4.77	+2.31	+6.54

Version max log MAP

En pratique, l'utilisation des logarithmes de probabilité permet de simplifier les calculs.

1) calcul des métriques de branche $\ln \gamma$

Les métriques de branches $\ln \gamma_i^{pq}$ s'obtiennent directement à partir des informations L_{INT}
$\ln \gamma_i^{00} = 0$
$\ln \gamma_i^{10} = L_{INT}(x_i^S)$
$\ln \gamma_i^{01} = L_{INT}(x_i^P)$
$\ln \gamma_i^{11} = L_{INT}(x_i^S) L_{INT}(x_i^P)$

$\ln \gamma_i^{00}$	0	0	0	0	0	0
$\ln \gamma_i^{10}$	+1.21	-1.00	-2.86	+4.84	+0.90	+5.87
$\ln \gamma_i^{01}$	+1.09	-1.07	+3.49	+0.72	+0.77	-1.13
$\ln \gamma_i^{11}$	+2.30	-2.07	+0.63	+5.56	+1.67	+4.74

De la même façon, au lieu de calculer les α et les β on calculera les $\ln \alpha$ et les $\ln \beta$

2) calcul des $\ln \alpha$

$i = 0$: initialisation

$\ln \alpha_0(0) = 0$
$\ln \alpha_0(1) = -\infty$

$i = 1$

$\ln \alpha_1(0) = \ln \alpha_0(0) + \ln \gamma_0^{00} = 0$
$\ln \alpha_1(1) = \ln \alpha_0(0) + \ln \gamma_0^{11} = 2.3$

$i = 2$

$\ln \alpha_2(0) = \max(\ln \alpha_1(0) + \ln \gamma_1^{00}; \ln \alpha_1(1) + \ln \gamma_1^{10}) = \max(0; 1.3) = 1.3$
$\ln \alpha_2(1) = \max(\ln \alpha_1(0) + \ln \gamma_1^{11}; \ln \alpha_1(1) + \ln \gamma_1^{01}) = \max(-2.07; 1.23) = 1.23$

$\vdots$

Les résultats finaux sont présentés sur la figure 1.11
3) calcul des $\ln \beta$

$i = 6$: initialisation

$\ln \beta_6(0) = 0$
$\ln \beta_6(1) = 0$ car le trellis n'a pas été terminé.

$i = 5$

$\ln \beta_5(0) = \max(\ln \beta_6(0) + \ln \gamma_5^{00}; \ln \beta_6(1) + \ln \gamma_5^{11} = \max(0; +4.74) = +4.74$

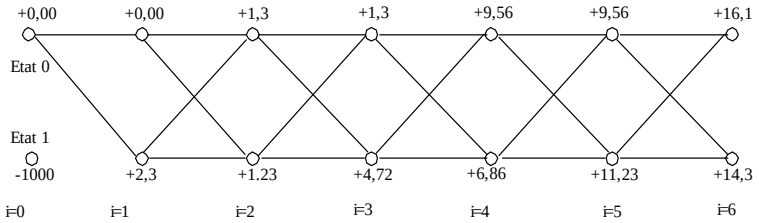


FIG. 1.11 – calcul des $\ln \alpha$

$\ln \beta_5(1) = \max(\ln \beta_6(0) + \ln \gamma_5^{10}; \ln \beta_6(1) + \ln \gamma_5^{01}) = \max(+5.87; -1.13) = +5.87$
$\vdots$

Les résultats finaux sont présentés sur la figure 1.12

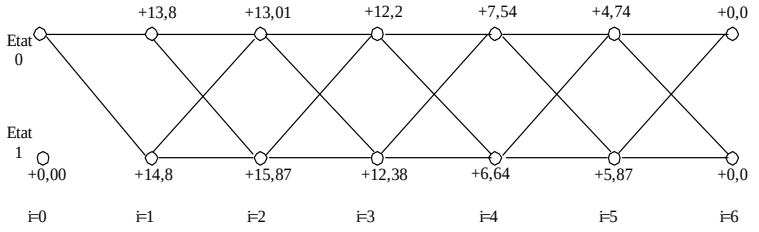


FIG. 1.12 – calcul des $\ln \beta$

4) calcul de $L_{APP}(x_i^S)$

On utilise la relation

$$L_{APP}(x_i^S) = \max_{m',m/x_i^S=+1} \left(\ln \alpha_t(m') + \ln \gamma_t(m',m) + \ln \beta_{t+1}(m) \right) - \max_{m',m/x_i^S=-1} \left(\ln \alpha_t(m') + \ln \gamma_t(m',m) + \ln \beta_{t+1}(m) \right) \quad (1.60)$$

$i = 0$

$$\begin{aligned} L_{APP}(x_0^S) =& (\ln \alpha_0(0) + \ln \gamma_0^{11} + \ln \beta_1(1)) \\ & - (\ln \alpha_0(0) + \ln \gamma_0^{00} + \ln \beta_1(0)) \\ & = (0 + 2.3 + 14.8) - (0 + 0 + 13.8) = 3.3 \end{aligned}$$

$i = 1$

$$\begin{aligned} L_{APP}(x_1^S) &= \max(\ln\alpha_1(0) + \ln\gamma_1^{11} + \ln\beta_2(1); \ln\alpha_1(1) + \ln\gamma_1^{10} + \ln\beta_2(0)) \\ &\quad - \max(\ln\alpha_1(0) + \ln\gamma_1^{00} + \ln\beta_2(0); \ln\alpha_1(1) + \ln\gamma_1^{01} + \ln\beta_2(1)) \\ &= \max(0 - 2.07 + 15.87; 2.3 - 1 + 13.01) \\ &\quad - \max(0 + 0 + 13.01; +2.3 - 1.07 + 15.87) \\ &= 14.31 - 17.1 = -2.79 \end{aligned}$$

$\vdots$

Finalement après calcul, on obtient les informations *a posteriori* suivantes :

$L_{APP}(x_i^S)$	+3.3	-2.79	-2.79	+5.02	+2.8	+7

Ainsi le décodeur max log MAP donne des valeurs sensiblement différentes par rapport au décodeur MAP. Ces différences proviennent de l'approximation max. En pratique on préfére cependant utiliser le décodeur max log MAP car il est plus facile à mettre en oeuvre.

1.4.3 Etude de la complexité de l'algorithme Aller-Retour

Pour étudier la complexité de l'algorithme Aller-Retour nous prendrons le cas du décodage d'un code convolutif sans connaissance des probabilités *a priori* sur les bits d'information.

Le nombre de multiplications et d'additions est donné pour une section du treillis. Pour un bloc de longueur K, il faudra donc multiplier les résultats par K pour obtenir le nombre total d'opération.

- **calcul des γs**
  Au lieu de calculer l'expression $\prod_{j=1}^{n} p(y_t^j \mid x_t^j(m', m))$ à chaque fois, il est préférable de calculer ses 2^n valeurs possibles et de les stocker en mémoire. Chaque calcul requiert $n-1$ multiplications élémentaires. Il faut donc effectuer $2^n \times (n-1)$ multiplications élémentaires pour le calcul des γs.
- **calcul des αs**
  Il y a 2^k branches quittant et arrivant sur chaque nœud. Pour chaque nœud, le calcul de α nécessite donc 2^k multiplications élémentaires et une addition de 2^k valeurs (soit $2^k - 1$ additions élémentaires). Puisqu'il y 2^M nœuds, le calcul des αs requiert donc 2^{k+M} multiplications et $2^M \times (2^k - 1)$ additions élémentaires.
- **calcul des βs**
  même complexité que pour le calcul des αs
- **calcul de** $APP(x_t^i = a)$

  Chaque calcul nécessite une addition de $2^{M-1} \times 2^k$ membres et $2^k \times 2^M$ multiplications. Comme il y a 2 valeurs à calculer ($APP(x_t^i = -1)$ et $APP(x_t^i = +1)$), il faut donc $2 \times ((2^{M-1} \times 2^k) - 1)$ additions élémentaires et $2 \times 2^M \times 2^k$ multiplications pour le calcul des APPs.

1.4.4 Lien entre l'algorithme Aller-Retour et l'algorithme Somme-Produit

Dans ce paragraphe, nous allons montrer que l'application de l'algorithme Somme-Produit sur le graphe factorisé d'un codeur convolutif permet de retrouver l'algorithme Aller-Retour.

Considérons le cas d'un système de transmission composé d'un codeur convolutif récursif systématique de rendement $R = 1/2$ terminé vu comme un code linéaire en bloc binaire systématique $(2K, K)$, et d'un canal discret stationnaire sans mémoire de densité de probabilité conditionnelle $p(y/x)$ comme montré sur la figure 1.13.

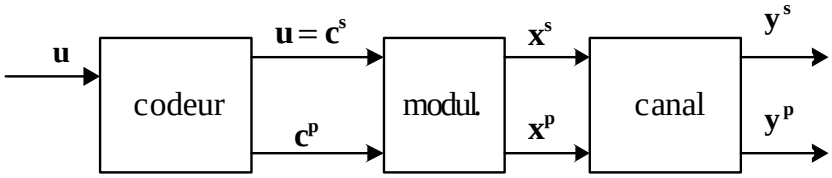


FIG. 1.13 – Modèle de Communication.

Soit $\mathbf{x} = (\mathbf{x^S}, \mathbf{x^P})$ la séquence émise sur le canal associée à la séquence d'entrée $\mathbf{u}$ avec $\mathbf{x^S} = (x_0^S, x_1^S, \ldots, x_i^S, \ldots, x_{K-1}^S)$ et $\mathbf{x^P} = (x_0^P, x_1^P, \ldots, x_i^P, \ldots, x_{K-1}^P)$. Soit $\mathbf{y} = (\mathbf{y^S}, \mathbf{y^P})$ la séquence reçue.

Nous avons vu que la probabilité conditionnelle *a posteriori* $APP(x_i^S = a)$ peut s'exprimer comme suit :

$$APP(x_i^S = a) = Pr\{x_i^S = a \mid \mathbf{y}\} \propto \sum_{\mathbf{x^S}: x_i^S = a} Pr\{\mathbf{x}\} p\{\mathbf{y}|\mathbf{x}\} \tag{1.61}$$

En considérant que la distribution *a priori* des séquences transmises sur le canal est uniforme, la probabilité qu'une séquence particulière $\mathbf{x}$ soit générée est égale à $1/2^K$.

Comme pour les graphes TWL, la fonction locale $T_i(s_i, c_i^P, c_i^S, s_{i+1})$ à la position i est définie à partir des équations d'état et de sortie du codeur convolutif.

$$T_i(s_i, c_i^P, c_i^S, s_{i+1}) = \begin{cases} 1 & \text{si les équations de parité sont valides} \\ 0 & \text{sinon} \end{cases} \tag{1.62}$$

Une séquence particulière $\mathbf{x}$ implique que le produit des K fonctions locales relatives à cette séquence est égal à 1. On a donc :

$$Pr\{\mathbf{x}\} = \frac{1}{2^K} \prod_{i=0}^{K-1} T_i(s_i, c_i^P, c_i^S, s_{i+1}) \tag{1.63}$$

La factorisation de $Pr\{\mathbf{x}\}p\{\mathbf{y}|\mathbf{x}\}$ est alors la suivante :

$$Pr\{\mathbf{x}\}p\{\mathbf{y}|\mathbf{x}\} = \frac{1}{2^K} \prod_{i=0}^{K-1} T_i(s_i, c_i^P, c_i^S, s_{i+1}) \prod_{i=0}^{K-1} p(y_i^S \mid x_i^S) \prod_{i=0}^{K-1} p(y_i^P \mid x_i^P) \tag{1.64}$$

Un exemple de graphe factorisé pour un codeur RSC de rendement 1/2 est donné en figure 1.14.

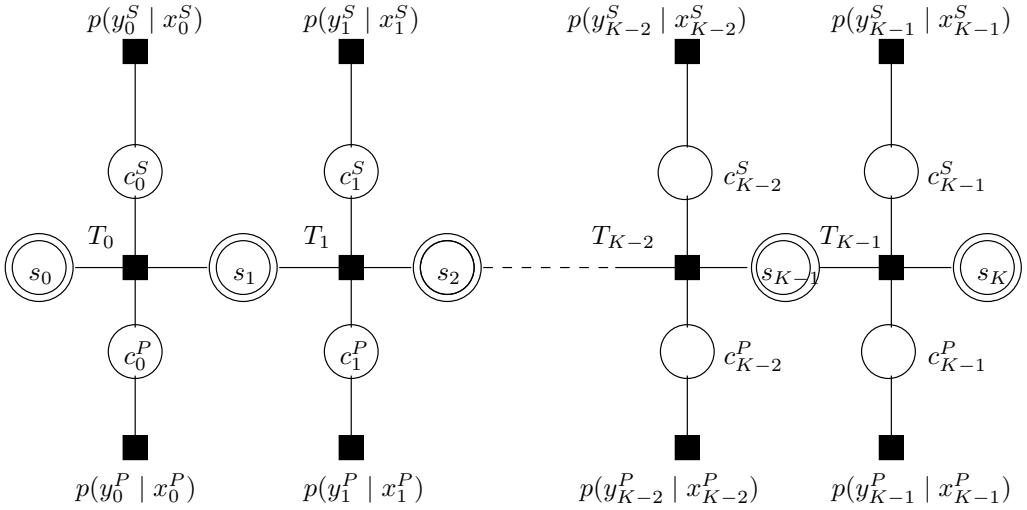


FIG. 1.14 – Exemple de graphe factorisé pour un codeur RSC de rendement 1/2 utilisé pour le décodage Aller-Retour.

Comme le graphe factorisé du codeur convolutif est sans cycle, nous pouvons appliquer l'algorithme Somme-Produit pour calculer les APPs.

Nous allons utiliser l'exemple de la figure 1.14 pour mettre en évidence que l'application de l'algorithme Somme-Produit sur ce graphe permet de retrouver exactement l'algorithme Aller-Retour.

Initialisation et calcul des γs : Comme pour tout graphe factorisé sans cycle, l'algorithme Somme-Produit commence aux extrémités du graphe. Chaque nœud de facteur $p(y_i^P|x_i^P)$ et $p(y_i^S|x_i^S)$ envoie un message vers le nœud de variable correspondant c_i^P et c_i^S. Comme ces nœuds de variable sont de degré 2, le message est directement transmis vers le nœud de contrôle correspondant. Ces messages $\mu_{c_i^P \to T_i}(c_i^P)$ et $\mu_{c_i^S \to T_i}(c_i^S)$ correspondent respectivement à $\gamma(c_i^P)$ et $\gamma(c_i^S)$ dans l'algorithme Aller-Retour. Les nœuds de variable d'état s_0 et s_K aux extrémités du graphe envoient respectivement au nœud de contrôle T_0 et T_{K-1} des messages triviaux $\mu_{s_0 \to T_0}(s_0)$ et $\mu_{s_K \to T_{K-1}}(s_K)$. Ces messages sont notés $\alpha(s_0)$ et $\beta(s_K)$ dans l'algorithme Aller-Retour .

Procédure aller : Le nœud de contrôle T_0 ayant reçu les messages de 3 de ses 4 branches, il transmet alors le message $\alpha(s_1)$ vers le nœud de variable d'état voisin s_1. Ce message est transmis au nœud de contrôle suivant T_1 . Cette procédure est répétée jusqu'à ce que le dernier nœud de variable s_K ait reçu le message $\alpha(s_K)$. Le calcul de $\alpha(s_{i+1})$ est le suivant :

$$\alpha(s_{i+1}) = \sum_{s_i} \sum_{c_i^P} \sum_{c_i^S} T_i(s_i, c_i^S, c_i^P, s_{i+1}) \alpha(s_i) \gamma(c_i^P) \gamma(c_i^S) \tag{1.65}$$

Procédure retour : La même procédure est effectuée dans le sens retour. Le calcul de $\beta(s_i)$ est le suivant :

$$\beta(s_i) = \sum_{s_i+1} \sum_{c_i^P} \sum_{c_i^S} T_i(s_i, c_i^S, c_i^P, s_{i+1}) \beta(s_{i+1}) \gamma(c_i^P) \gamma(c_i^S) \tag{1.66}$$

Calcul des probabilités extrinsèques : Finalement, il ne reste plus qu'à transmettre les messages $\mu_{T_i \to c_i^S}(c_i^S)$ entre les nœuds de contrôle T_i et les nœuds de variable u_i.

$$EXTR(x_i^S) = \sum_{s_i} \sum_{s_{i+1}} \sum_{c_i^P} T_i(s_i, c_i^S, c_i^P, s_{i+1}) \alpha(s_i) \beta(s_{i+1}) \gamma(c_i^P) \tag{1.67}$$

Pour chacune de ces sommes, $T_i(s_i, c_i^S, c_i^P, s_{i+1}) = 0$ sauf lorsque la branche du treillis est valide.

Calcul des probabilités *a posteriori* : Le calcul de $APP(x_i^S)$ s'obtient à partir de $EXTR(x_i^S)$ et $\gamma(c_i^S)$ comme suit :

$$APP(x_i^S) = EXTR(x_i^S) \times \gamma(c_i^S) \tag{1.68}$$

Les messages $\alpha(s_i)$ et $\beta(s_i)$ ont une interprétation en terme de probabilité. Pour chaque valeur de $s_i \in S_i$, $\alpha(s_i)$ est proportionnel à la probabilité conditionnelle que la séquence émise passe par l'état s_i conditionnellement à l'observation passée $\mathbf{y}_0, \ldots, \mathbf{y}_{i-1}$.

$$\alpha(s_i) \propto Pr\{s_i = S_i | \mathbf{y}_0^{i-1}\} \tag{1.69}$$

Pour chaque valeur de $s_{i+1} \in S_{i+1}$, $\beta(s_{i+1})$ est proportionnel à la probabilité conditionnelle que la séquence émise passe par l'état s_{i+1} sachant l'observation future $\mathbf{y}_{i+1}, \mathbf{y}_{i+2}, \ldots$.

$$\beta(s_{i+1}) \propto Pr\{s_{i+1} = S_{i+1} | \mathbf{y}_{i+1}^{K-1}\} \tag{1.70}$$

On retrouve bien les expressions $\alpha(s_i)$ et $\beta(s_{i+1})$ de l'algorithme Aller-Retour présentées précédemment.

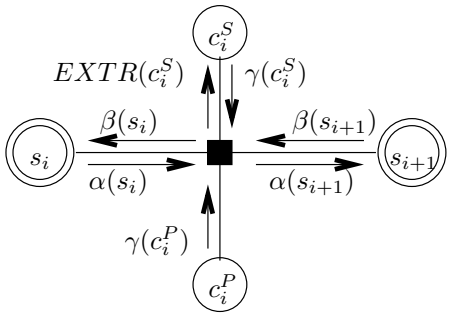


FIG. 1.15 – Passage des messages lors de l'application de l'algorithme Aller-Retour.

1.5 Codes PCBC

1.5.1 Structure

La structure des codes linéaires en blocs concaténés en parallèle PCBC (*parallel concatenated block codes* en anglais) est donnée sur la figure 1.16.

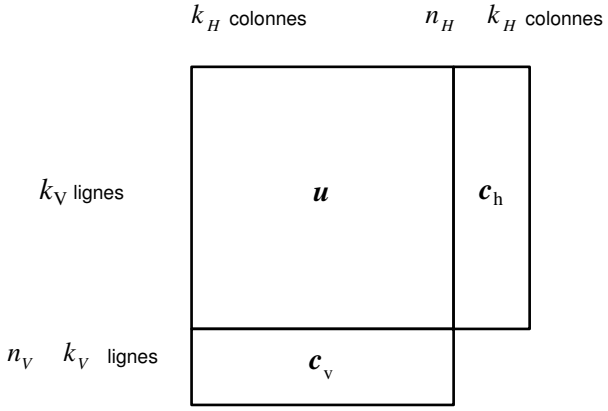


FIG. 1.16 – Code PCBC.

Les symboles d'information sont disposés sous la forme d'un tableau de dimension $k_V k_H$. A ce tableau, on a ajouté deux tableaux de dimension $k_V(n_H - k_H)$ et $(n_V - k_V)k_H$. Le code PCBC ainsi obtenu est un code linéaire en bloc systématique (N, K) avec $N = k_V n_H + n_V k_H - k_V k_H$ et $K = k_V k_H$. Le rendement global du code PCBC est donc égal à :

$$R_T = \frac{k_V k_H}{k_V n_H + n_V k_H - k_V k_H}$$

Chacune des k_V lignes correspond à un code en bloc linéaire systématique C_H (n_H, k_H). Par ailleurs, chaque colonne correspond également à un code en bloc linéaire systématique C_V (n_V, k_V).

La distance minimale de ces codes est égale à la somme $d_V + d_H - 1$ où d_V et d_H sont respectivement la distance minimale des codes constituants C_V et C_H.

Le graphe TWL des codes PCBC est présenté sur la figure 1.17.

Dans ce chapitre nous limiterons notre étude aux codes PCBC binaires. Ces codes peuvent être décodés simplement de façon itérative.

Les codes constituants sont principalement des codes de parité, des codes de Hamming ou des codes BCH.

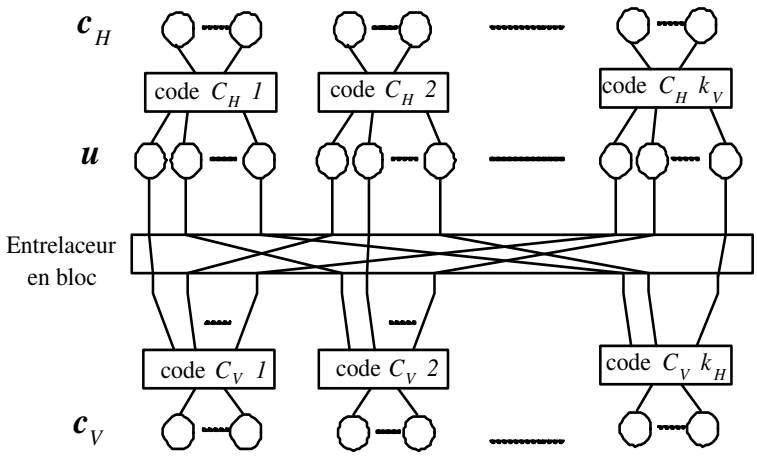


FIG. 1.17 – Graphe TWL des codes PCBC.

1.5.2 Décodage itératif

Pour que le décodage itératif soit efficace, le graphe TWL du code concaténé ne doit pas comporter de cycles courts. C'est le rôle de l'entrelaceur en bloc ligne-colonne.

Les étapes du décodage itératif sont les suivantes :

- étape 1 : initialisation des informations *a priori* $L^H_{APRI} = 0$ pour le décodage horizontal.
- étape 2 : décodage horizontal. Calcul des informations extrinsèques $L^H_{EXTR}(x_i)$ associés aux bits d'information u_i
- étape 3 : les informations extrinsèques calculées $L^H_{EXTR}(x_i)$ deviennent les informations *a priori* $L_{APRI}(x_i)$ pour le décodage vertical.

$$L_{APRI}(x_i) = L^H_{EXTR}(x_i)$$

- étape 4 : décodage vertical. Calcul des informations extrinsèques $L^V_{EXTR}(x_i)$ associés aux bits d'information u_i
- étape 5 : si le nombre d'itérations (décodage horizontal puis vertical) fixés est atteint on termine le décodage par l'étape 6 sinon $L_{APRI}(x_i) = L^V_{EXTR}(x_i)$ puis retour à l'étape 2
- étape 6 : calcul de l'information *a posteriori* $L_{APP}(x_i) = L_{APRI}(x_i) + L_{INTR}(x_i)) + L^V_{EXTR}(x_i)$

Si le graphe de Tanner du code constituant est un arbre, on appliquera l'algorithme Somme-Produit ou Minimum-Somme sur cet arbre pour calculer les informations extrinsèques.

Si le graphe de Tanner du code constituant n'est pas un arbre on le transformera en un graphe TWL sans cycle (donc un arbre) et on appliquera l'algorithme Aller-Retour.

Une autre solution proposée par Pyndiah [39] consiste à utiliser l'algorithme de Chase [12]. Cet algorithme permet de réaliser un décodage à entrées et sorties pondérées à partir d'un algorithme de décodage à entrées et sorties dures comme l'algorithme de Berlekamp-Massey.

exemple [45] : considérons le code PCBC $(8, 4)$ défini par les 4 équations de parité suivantes :

$$\begin{aligned} u_1 + u_2 + c_5 &= 0 \qquad T_1 \\ u_3 + u_4 + c_6 &= 0 \qquad T_2 \\ u_1 + u_3 + c_7 &= 0 \qquad T_3 \\ u_2 + u_4 + c_8 &= 0 \qquad T_4 \end{aligned} \tag{1.71}$$

La structure de ce code PCBC est la suivante :

u_1	u_2	c_5
u_3	u_4	c_6
c_7	c_8	

Le graphe de Tanner associé est présenté sur la figure 1.18.

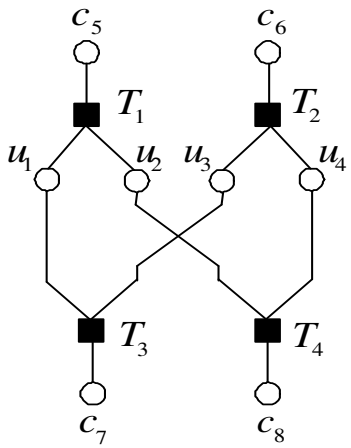


FIG. 1.18 – Graphe de Tanner du code PCBC (8,4).

Dans cet exemple, comme le graphe de Tanner du code constituant est un arbre, on utilisera l'algorithme Minimum-Somme pour le calcul des informations extrinsèques.

Considérons le mot d'information binaire suivant :

$$\mathbf{u} = [1001]$$

Le mot émis est égal à :

$$\mathbf{x} = [+1 - 1 - 1 + 1 + 1 + 1 + 1 + 1]$$

Considérons le mot reçu $\mathbf{y}$ suivant ($\mathbf{y} = \mathbf{x} + \mathbf{n}$) en sortie du canal à bruit blanc additif gaussien de variance $\sigma^2 = 1$:

$$\mathbf{y} = [+0.75 + 0.05 + 0.1 + 0.15 + 1.25 + 1.0 + 3.0 + 0.5]$$

Les informations intrinsèques sont calculées en utilisant la relation $L_{INT}(x_i) = \frac{2y_i}{\sigma^2}$:

$$\{L_{INT}(x_i)\} = +1.5 + 0.1 + 0.2 + 0.3 + 2.5 + 2.0 + 6.0 + 1.0$$

mot de code binaire

$u_1 = 1$	$u_2 = 0$	$c_5 = 1$
$u_3 = 0$	$u_4 = 1$	$c_6 = 1$
$c_7 = 1$	$c_8 = 1$	

mot émis sur le canal

$x_1 = +1$	$x_2 = -1$	$x_5 = +1$
$x_3 = -1$	$x_4 = +1$	$x_6 = +1$
$x_7 = +1$	$x_8 = +1$	

mot reçu

$y_1 = 0.75$	$y_2 = 0.05$	$y_5 = 1.25$
$y_3 = 0.1$	$y_4 = 0.15$	$y_6 = 1.0$
$y_7 = 3.0$	$y_8 = 0.05$	

TAB. 1.1 – Exemple code PCBC (8,4)

+1.5	+0.1	+2.5
+0.2	+0.3	+2.0
+6.0	+1.0	

TAB. 1.2 – Informations intrinsèques $L_{INT}(x_i)$

A chaque itération on déterminera tout d'abord les informations extrinsèques $L^H_{EXTR}(x_i)$ associées aux bits d'information en utilisant la relation (1.29) (décodage horizontal) :

$$L^H_{EXTR}(x_1) = -(L_{APRI}(x_2) + L_{INTR}(x_2)) \boxplus L_{INTR}(x_5)$$
$$L^H_{EXTR}(x_2) = -(L_{APRI}(x_1) + L_{INTR}(x_1)) \boxplus L_{INTR}(x_5)$$
$$L^H_{EXTR}(x_3) = -(L_{APRI}(x_4) + L_{INTR}(x_4)) \boxplus L_{INTR}(x_6)$$
$$L^H_{EXTR}(x_4) = -(L_{APRI}(x_4) + L_{INTR}(x_4)) \boxplus L_{INTR}(x_6)$$

On rappelle que l'opérateur $\boxplus$ a été défini pour simplifier l'écriture dans la suite du document. puis ensuite les informations $L^V_{EXTR}(x_i)$ (décodage vertical)

$$L^V_{EXTR}(x_1) = -(L_{APRI}(x_3) + L_{INTR}(x_3)) \boxplus L_{INTR}(x_7)$$
$$L^V_{EXTR}(x_3) = -(L_{APRI}(x_1) + L_{INTR}(x_1)) \boxplus L_{INTR}(x_7)$$
$$L^V_{EXTR}(x_2) = -(L_{APRI}(x_4) + L_{INTR}(x_4)) \boxplus L_{INTR}(x_8)$$
$$L^V_{EXTR}(x_4) = -(L_{APRI}(x_2) + L_{INTR}(x_2)) \boxplus L_{INTR}(x_8)$$

Analysons le décodage itératif pour les valeurs de l'exemple proposé.

Première itération (décodage horizontal) :

Il n'y a pas d'information *a priori* donc $L_{APRI}(x_i) = 0$

$$L^H_{EXTR}(x_1) = -(0 + 0.1) \boxplus 2.5 = -0.1$$
$$L^H_{EXTR}(x_2) = -(0 + 1.5) \boxplus 2.5 = -1.5$$

$$L^H_{EXTR}(x_3) = -(0 + 0.3) \boxplus 2.0 = -0.3$$
$$L^H_{EXTR}(x_4) = -(0 + 0.2) \boxplus 2.0 = -0.2$$

Première itération (décodage vertical) :

Information *a priori* $L_{APRI}(x_i) = L^H_{EXTR}(x_i)$ pour $i = 1, 2, 3$ et 4

$$L^V_{EXTR}(x_1) = -(-0.3 + 0.2) \boxplus 6.0 = 0.1$$
$$L^V_{EXTR}(x_3) = -(-0.1 + 1.5) \boxplus 6.0 = -1.4$$
$$L^V_{EXTR}(x_2) = -(-0.2 + 0.3) \boxplus 1.0 = -0.1$$
$$L^V_{EXTR}(x_4) = -(-1.5 + 0.1) \boxplus 1.0 = 1.0$$

Les résultats sont synthétisés dans les tables suivantes :

1.5	0.1
0.2	0.3

(a)

-0.1	-1.5
-0.3	-0.2

(b)

0.1	-0.1
-1.4	1.0

(c)

TAB. 1.3 – Informations associées aux bits d'information en fin de première itération(a) $L_{INTR}(x_i)$ (b) $L_{APRI}(x_i)$ (c) $L^V_{EXTR}(x_i)$

Il est possible de calculer les informations *a posteriori* associées aux bits d'information en fin de première itération en utilisant la relation $L_{APP}(x_i) = L_{APRI}(x_i) + L_{INTR}(x_i)) + L^V_{EXTR}(x_i)$

1.5	-1.5
-1.5	1.1

TAB. 1.4 – Calcul des informations *a posteriori*$L_{APP}(x_i)$ associées aux bits d'information en fin de première itération

Sur cet exemple, on voit qu'une seule itération suffit pour décoder correctement le mot reçu. Poursuivons par une seconde itération afin d'améliorer les informations *a posteriori*.

Seconde itération (décodage horizontal) :

$L_{APRI}(x_i) = L^V_{EXTR}(x_i)$ pour $i = 1, 2, 3$ et 4

$$L^H_{EXTR}(x_1) = -(-0.1 + 0.1) \boxplus 2.5 = 0.0$$
$$L^H_{EXTR}(x_2) = -(0.1 + 1.5) \boxplus 2.5 = -1.6$$
$$L^H_{EXTR}(x_3) = -(1.0 + 0.3) \boxplus 2.0 = -1.3$$
$$L^H_{EXTR}(x_4) = -(-1.4 + 0.2) \boxplus 2.0 = 1.2$$

Seconde itération (décodage vertical) :

$L_{APRI}(x_i) = L^H_{EXTR}(x_i)$ pour $i = 1, 2, 3$ et 4

$$L^V_{EXTR}(x_1) = -(-1.3 + 0.2) \boxplus 6.0 = 1.1$$
$$L^V_{EXTR}(x_3) = -(0.0 + 1.5) \boxplus 6.0 = -1.5$$

$$L^V_{EXTR}(x_2) = -(1.2 + 0.3) \boxplus 1.0 = -1.0$$
$$L^V_{EXTR}(x_4) = -(-1.6 + 0.1) \boxplus 1.0 = 1.0$$

Les résultats sont à nouveau synthétisés dans les tables suivantes :

1.5	0.1
0.2	0.3

(a)

0.0	-1.6
-1.3	1.2

(b)

1.1	-1.0
-1.5	1.0

(c)

TAB. 1.5 – Informations en fin de seconde itération(a) $L_{INTR}(x_i)$ (b) $L_{APRI}(x_i)$ (c) $L^V_{EXTR}(x_i)$

Il est possible de calculer les informations *a posteriori* en fin de seconde itération en utilisant la relation $L_{APP}(x_i) = L_{APRI}(x_i) + L_{INTR}(x_i)) + L^V_{EXTR}(x_i)$

2.6	-2.5
-2.6	2.5

TAB. 1.6 – Calcul des informations *a posteriori* $L_{APP}(x_i)$ en fin de seconde itération

Les fiabilités *a posteriori* $L_{APP}(x_i)$ sur les bits d'information obtenues en fin de seconde itération sont meilleures. Cependant les informations *a priori* sont de plus en plus corrélées et une troisième itération n'apportera pas de gain supplémentaire.

En pratique, 5 à 6 itérations suffisent pour obtenir des performances proches de celles d'un décodeur à maximum de vraisemblance.

1.6 Codes LDPC

Les codes LPDC (*low density parity check codes* en anglais) ont été introduits par Gallager en 1963 [19] et ont été redécouverts en 1996 par MacKay [31].

1.6.1 Définition

Un code LDPC (N, d_c, d_T) est un code linéaire binaire défini par sa matrice de parité $\mathbf{H}$ de dimension $M \times N$. Cette matrice est de faible densité c'est-à-dire que le nombre de "1" sur chacune de ses lignes est petit devant la taille du bloc de bits informations. Chaque ligne de la matrice $\mathbf{H}$ contient d_T "1" et chaque colonne contient d_c "1" et des "0" partout ailleurs. La matrice de parité décrit les M équations de parité du code. On a

$$M = N\frac{d_c}{d_T}$$

Le graphe de Tanner associé à ce code est biparti. Ce graphe est dit régulier si les nœuds de variable ont tous le même degré d_c et si les nœuds de contrôle ont aussi tous le même degré d_T. Le graphe de Tanner d'un code LDPC est donné sur la figure 1.19. La matrice de parité associée est la suivante :

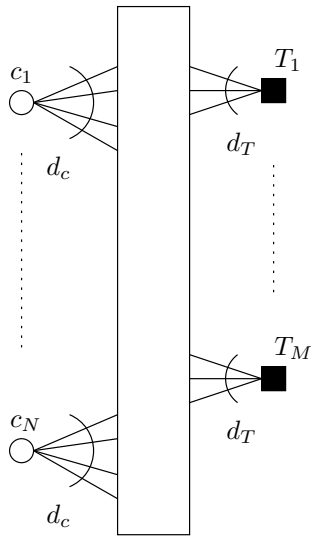


FIG. 1.19 – Graphe de Tanner d'un code LDPC (N, d_c, d_T) régulier.

Le rendement $R = \frac{K}{N}$ du code LDPC est

$$R \geq 1 - \frac{d_c}{d_T} \tag{1.72}$$

On a l'égalité lorsque toutes les lignes de la matrice $\mathbf{H}$ sont indépendantes. Dans ce cas, la taille du mot d'information est alors égale à $K = N(1 - \frac{d_c}{d_T})$.

exemple. Soit le code LDPC (12,3,4) dont le graphe de Tanner est donné sur la figure 1.20. La matrice de parité associée à ce code est la suivante :

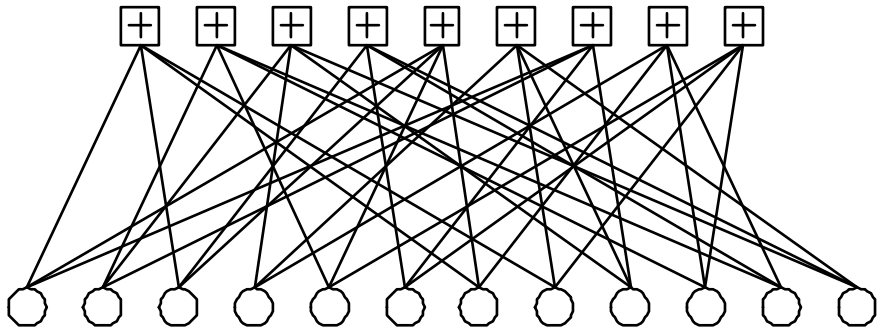


FIG. 1.20 – Graphe de Tanner d'un code LDPC $(12, 3, 4)$ régulier.

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{1.73}$$

Comme les lignes de la matrice $\mathbf{H}$ sont indépendantes, le rendement de ce code est égal à $R = \frac{1}{4}$, et la longueur du mot d'information est $K = 3$.

Gallager a démontré que la distance minimale moyenne calculée sur l'ensemble des codes LDPC croît linéairement avec K lorsque $d_c > 2$ et donc que les codes LDPC avec $d_c > 2$ sont asymptotiquement bons (ils permettent d'atteindre la borne de Gilbert-Varshamov) . Par contre, pour $d_c = 2$, la distance croit seulement linéairement avec le logarithme de K (distance en $\mathbb{O}(\log K)$).

Différents travaux de recherche ont montré que les codes LDPC irréguliers permettent de s'approcher de la limite de Shannon pour une longueur de mot de code très grande [28]. Pour les codes LDPC irréguliers, les nœuds de variable et de contrôle sont de différents degrés.

Il existe également des méthodes dont nous ne discuterons pas ici qui permettent de rendre la complexité du codeur LDPC linéaire avec la taille K (complexité en $\mathbb{O}(K)$).

1.6.2 Décodage à décisions dures des codes LDPC

La complexité d'un décodeur au maximum de vraisemblance pour les codes LDPC croît exponentiellement avec la taille du mot d'information K. Heureusement, il existe des solutions sous optimales mais de complexité raisonnable : il s'agit de décodeurs itératifs. Nous allons tout d'abord présenter les algorithmes A et B proposés par Gallager [19].

Description de l'algorithme A de Gallager

On considère la transmission d'un mot de code LDPC sur un canal BSC. Soit y_m le bit reçu (0 ou 1) associé au nœud de variable c_m.

Définissons $\mu^i_{c_n \to T_m}(c_n)$ le bit transmis du nœud de variable c_n vers le nœud de contrôle T_m et soit $\mu^i_{T_m \to c_n}(c_n)$ le bit transmis du nœud de contrôle T_m vers le nœud de variable c_n à l'itération i.

Pour chaque branche (c_n, T_m) :
- **Mise à jour des bits** $\mu^i_{c_n \to T_m}(c_n)$:
  A l'itération 0, $\mu^0_{c_n \to T_m}(c_n) = y_m$
  A l'itération i avec $i > 0$,
  Si tous les bits $\mu^{i-1}_{T_{m'} \to c_n}(c_n) = b$ où $T_{m'}$ sont les nœuds de controle reliés à c_n excepté T_m(soit $d_c - 1$ bits) alors $\mu^i_{c_n \to T_m}(c_n) = b$
  Sinon, $\mu^i_{c_n \to T_m}(c_n) = y_m$
- **Mise à jour des bits** $\mu^i_{T_m \to c_n}(c_n)$:
  $\mu^i_{T_m \to c_n}(c_n)$ est la somme modulo 2 des bits $\mu^i_{c_{n'} \to T_m}(c_{n'})$ où $c_{n'}$ sont les nœuds de variable reliés à T_m excepté c_n(soit $d_T - 1$ bits)

Comme nous le verrons par la suite, les bits $\mu^i_{c_n \to T_m}(c_n)$ et $\mu^i_{T_m \to c_n}(c_n)$ correspondent à des informations extrinsèques. Ce sont des estimations du bit c_m à l'itération i.

Description de l'algorithme B de Gallager

Dans l'algorithme A, pour avoir $\mu^i_{c_n \to T_m}(c_n) = b$, il est nécessaire que tous les bits $\mu^{i-1}_{T_{m'} \to c_n}(c_n)$ soient égaux à b.

Dans l'algorithme B, cette contrainte est relaxée : si le nombre de bits $\mu^{i-1}_{T_{m'} \to c_n}(c_n) = b$ est supérieur à un seuil S, alors $\mu^i_{c_n \to T_m}(c_n) = b$. Le seuil S est choisi afin de minimiser la probabilité d'erreur bit.

Ces deux algorithmes ne sont efficaces que si le graphe de Tanner des codes LDPC ne présentent pas de cycle. Bien que cela ne soit jamais le cas, différents travaux ont montré que sous réserve que la taille des blocs soit suffisamment grande, ces cycles n'influencent pas ou peu la convergence de l'algorithme.

1.6.3 Décodage itératif des codes LDPC (d_c, d_T) réguliers

On considère maintenant une modulation bipodale (BPSK) et un canal BBAG ($\mathbf{y} = \mathbf{x} + \mathbf{n}$) avec $x_i = \pm 1$ et n_i échantillon de bruit blanc additif gaussien centré de variance σ^2.

On supposera que la taille des mots de code est très grande. En optimisant la construction de l'entrelaceur, les cycles dans le graphe de Tanner seront alors très grands et donc sans influence sur le passage de l'information extrinsèque pendant le décodage itératif.

Nous allons utiliser le graphe factorisé du code LPDC présenté sur la figure 1.21 pour décrire l'algorithme de décodage itératif. Ce graphe factorisé se déduit du graphe de Tanner du code LDPC.

Le décodage d'un code LPDC consiste à appliquer un algorithme de décodage à décisions pondérées comme l'algorithme Somme-Produit sur le graphe de Tanner du code. Richardson et Urbanke [41] ont démontré que pour des tailles de bloc d'information K très grandes, les messages se propagent sur un arbre avec une probabilité $1 - \mathbb{O}(\frac{1}{K})$. Lorsque le graphe local utilisé pour le décodage est un arbre, nous pouvons analyser simplement l'algorithme de décodage car les messages qui arrivent à chaque nœud sont alors indépendants (chaque nœud de variable n'apparaît qu'une seule fois sur cet arbre local).

Sur la figure 1.22, nous présentons un exemple d'arbre local pour un code LDPC (d_c, d_T). Sur cet arbre, nous pouvons observer que chaque nœud de contrôle est défini par un simple code de parité $(d_T, d_T - 1)$.

L'application de l'algorithme Somme-Produit sur ce graphe consiste à propager des messages entre les nœuds de variables et les nœuds de contrôle itérativement.

Comme pour les codes PCBC, une itération comprend deux phases :

-calcul des messages nœud de variable vers nœud de contrôle : chaque nœud de variable v reçoit d_c messages issus des nœuds de contrôle voisins plus le message issu du canal de transmission, calcule les d_c messages puis envoie ceux-ci aux nœuds de contrôle voisins.

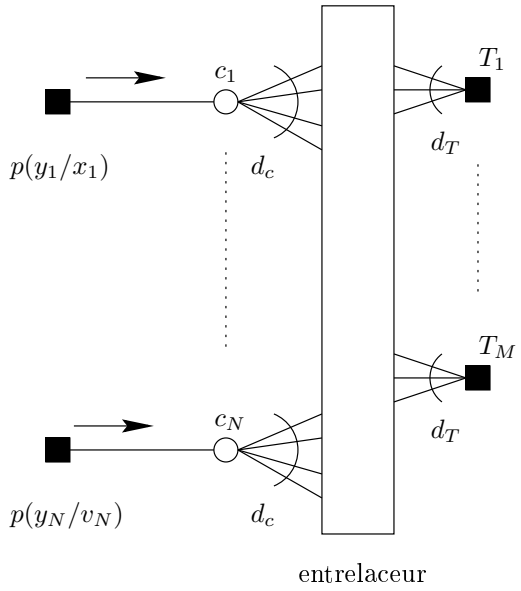


FIG. 1.21 – Graphe factorisé d'un code LDPC (N, d_c, d_T) régulier.

-calcul des messages nœud de contrôle vers nœud de variable : chaque nœud de contrôle c reçoit d_T messages issus des nœuds de variables voisins , calcule les d_T messages puis envoie ceux-ci aux nœuds de variables voisins.

Dans les deux phases, chaque message est calculé à partir de tous les messages entrant à l'exception du message issu de la branche considérée.

En utilisant les 2 règles de calcul présentées précédemment lorsque les messages sont des logarithmes de rapport de vraisemblance, nous obtenons les relations suivantes pour les messages $\mu_{c\to T}^{(l)}(c)$ et $\mu_{T\to c}^{(l)}(c)$:

-calcul des messages nœud de variable vers nœud de contrôle $\mu_{c\to T}^{(l)}(c)$:

$$\mu_{c\to T}^{(l)}(c) = \sum_{i=1}^{d_T-1} \mu_{T_i\to c}^{(l-1)}(c) + \mu_0(c) \tag{1.74}$$

Lors de la première itération aucune information *a priori* issue des précédents décodeurs n'étant disponible, seule l'information issue du canal de transmission μ_0 est utilisée dans le calcul de $\mu_{c\to T}(c)$.

$\mu_0(c)$ est le rapport de vraisemblance du symbole reçu :

$$\mu_0(c) = \ln \frac{p(y/x=+1)}{p(y/x=-1)} = \frac{2}{\sigma^2} y$$

-calcul des messages contrôle vers variable $\mu_{T\to c}^{(l)}(c)$:

$$\mu_{T\to c}^{(l)}(c) = -2\tanh^{-1}\left(\prod_{i=1}^{d_c-1} \tanh\left(\frac{\mu_{c_i\to T}^{(l)}(c_i)}{2}\right)\right) \tag{1.75}$$

Finalement, le calcul de l'information *a posteriori* $APP(x)$ associé au bit c est effectué en sommant les informations issues de tous les nœuds de contrôle et du canal :

$$APP(x) = \sum_{i=1}^{d_c} \mu_{T_i\to c}^{(l)}(c) + \mu_0(c) \tag{1.76}$$

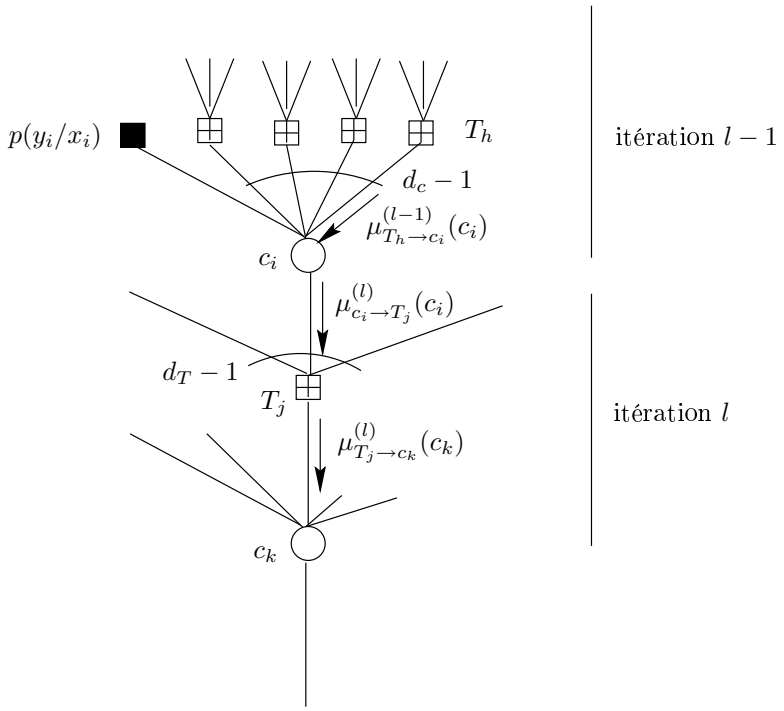


FIG. 1.22 – Exemple d'arbre local pour un code LDPC (d_c, d_T)

1.7 Codes convolutifs concaténés en parallèle ou Turbo codes

1.7.1 Structure générale

La structure du codeur associé au turbo code ou codes convolutifs concaténés en parallèle (PCC) (*parallel concatenated convolutional coders* en anglais) est présentée figure 1.24. Ce codeur comprend 2 codeurs convolutifs récursifs systématiques (RSC) binaires de rendement k/n et 1 entrelaceur de K bits. La longueur de la séquence d'entrée est limitée à K bits.

Le Turbo code ainsi construit est un code en bloc (N, K) avec $N = (2\frac{n-k}{k}+1)K$ en négligeant les éventuels bits de terminaison. Le rendement global du code est donc égal à :

$$R_T = \frac{1}{2\frac{n-k}{k}+1}$$

La fonction d'entrelacement appliquée sur les K bits d'information modifie l'ordre de sortie de ceux-ci.

Un entrelaceur E de taille K est défini par sa matrice de permutation Π de dimension $K \times K$; nous avons alors la relation entre la séquence entrée $\mathbf{u}$ et la sortie $\mathbf{v}$ de l'entrelaceur E :

$$\mathbf{v} = \mathbf{u}\Pi \qquad \text{avec} \quad \mathbf{u} = [u_0, u_1, \ldots, u_{K-1}]$$
$$\mathbf{v} = [v_0, v_1, \ldots, v_{K-1}]$$

$$\Pi = \{a_{ij}\}_{K\times K} \quad \text{avec} \quad a_{ij} \in \{0,1\}$$

$$\sum_j a_{ij} = 1 \quad \text{et} \quad \sum_i a_{ij} = 1$$

Si a_{ij}=1 alors le bit u_i est associé au bit v_j.

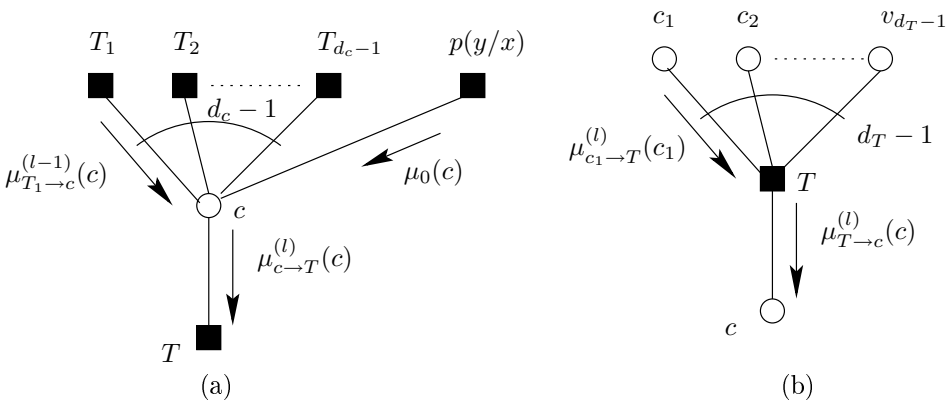


FIG. 1.23 – Flot de message pour les calculs de $\mu^{(l)}_{c\to T}(c)$ et $\mu^{(l)}_{T\to c}(c)$.

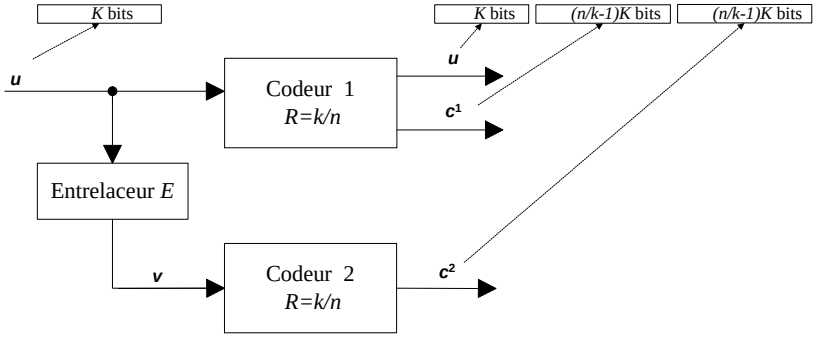


FIG. 1.24 – Codeurs convolutifs concaténés en parallèle

La structure initialement proposée par Berrou *et al.* comporte 2 codeurs RSC de rendement $R = 1/2$. Le rendement global est alors égal à $R_T = 1/3$. Le graphe TWL de ce codeur PCC de rendement $R_T = 1/3$ est donné figure 1.25.

Comme pour les codes convolutifs, il est possible d'effectuer une perforation en sortie afin d'augmenter le rendement global.

Les critères de construction de l'entrelaceur sont doubles :

Le premier critère de construction de l'entrelaceur est relatif à la distribution de poids des codes convolutifs concaténés en parallèle : l'entrelaceur doit être construit de manière à éviter les permutations qui entraînent des mots codes de faible poids. Il doit permettre d'améliorer la distribution de poids des codes concaténés et d'augmenter leurs distances minimales. La structure parallèle des codes PCC implique que le poids de Hamming du mot code est égal à la somme des poids de Hamming des séquences $\mathbf{u}$, $\mathbf{c}^1$ et $\mathbf{c}^2$. Ainsi l'entrelaceur doit entrelacer les séquences d'entrée $\mathbf{u}$ qui génèrent des séquences $\mathbf{c}^1$ de poids faible, avec les séquences $\mathbf{v}$ qui génèrent des séquences $\mathbf{c}^2$ de poids fort, et vice versa

Le second critère de construction est relatif au décodage itératif. L'entrelaceur doit garantir le meilleur passage de l'information extrinsèque d'un décodeur à l'autre. Autrement dit, l'entrelaceur doit permettre à chaque itération que l'information *a priori* soit la moins corrélée possible avec l'information issue du canal de transmission. L'objectif de ce critère est d'assurer que les performances du décodeur itératif soient aussi proches que possible de celles d'un décodeur à maximum de vraisemblance. Ce critère est satisfait sous réserve que le graphe TWL du code concaténé soit

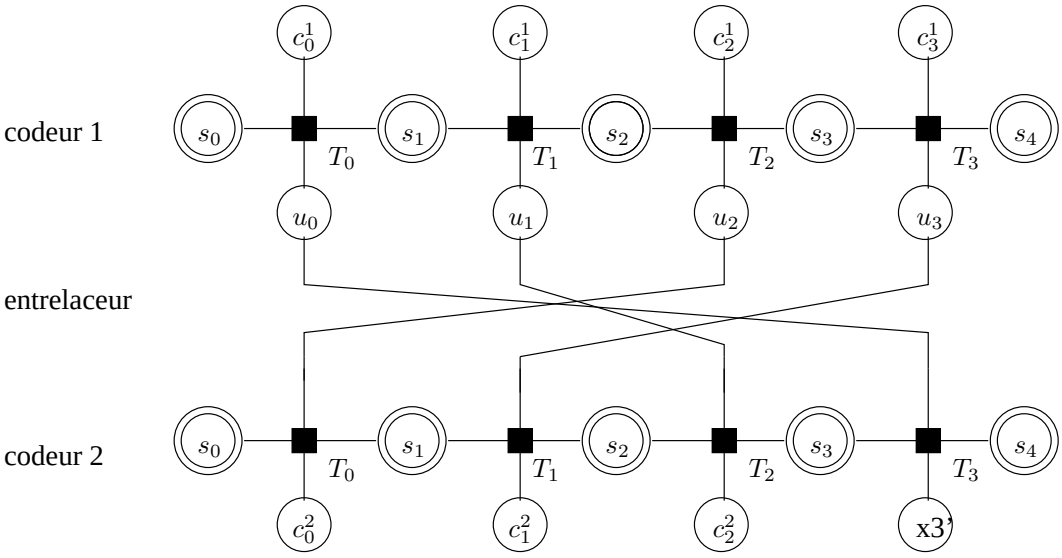


FIG. 1.25 – Graphe TWL d'un codeur PCC

sans cycle. Comme les cycles sont toujours présents, le second critère de construction doit garantir que ces cycles soient suffisamment grands.

Le rôle de la terminaison est d'éviter les mots de code de faible poids en ajoutant en fin de séquence M bits afin de ramener l'état interne du codeur convolutif à l'état zéro où M est le nombre de cellules mémoires du codeur convolutif. Pusieurs techniques de terminaison sont possibles comme la terminaison des 2 codeurs séparemment et la terminaison des 2 codeurs conjointement. Une autre approche consiste à ramener l'état du codeur dans l'état initial. La fin du treillis du code est alors rebouclée sur son début. Cette technique est appelée *tail-biting* en anglais.

1.7.2 Etude des performances moyennes des turbo codes

Dans [6], Benedetto et Montorsi ont introduit un entrelaceur uniforme pour étudier les performances moyennes des codes PCC. Un entrelaceur uniforme n'est pas un entrelaceur au sens strict mais plutôt un outil probabiliste : il associe une séquence d'entrée de poids w avec les $\binom{K}{w}$ séquences distinctes de poids w, chacune avec la même probabilité $p = 1/\binom{K}{w}$. Cet entrelaceur permet de calculer l'énumérateur de poids moyen des codes convolutifs concaténés. Nous avons introduit précedemment la fonction d'énumération de poids IRWEF $A^C(W,Z)$ d'un codeur binaire en bloc (N,K). La fonction IRWEF d'un codeur RSC C de rendement k/n dont la séquence d'entrée est terminée et de longueur K bits est la suivante :

$$A^C(W,Z) = \sum_{w=w_{min}}^{K} \sum_{z=z_{min}}^{\frac{n-k}{k}K} A^C_{w,z} W^w Z^z = \sum_{w=w_{min}}^{K} W^w A^C(w,Z)$$

$$\text{avec} \quad A^C(w,Z) = \sum_{z=z_{min}}^{\frac{n-k}{k}K} A^C_{w,z} Z^z$$

$A^C_{w,z}$ est le nombre de mots de code de C dont le poids de la séquence d'entrée est égal à w et dont le poids de la séquence des bits de redondance est égal à z. w_{min} est le poids minimal des séquences d'entrée. z_{min} est le poids minimum des bits de redondance. Le calcul de la fonction d'énumération de poids $A^C(W,Z)$ pour un code convolutif terminé est développé dans l'annexe 1.

Dans le cas des codeurs RSC terminés, on a $w_{min} = 2$. Nous avons alors la relation suivante entre d_{libre} la distance libre ou minimale du codeur RSC et z_{min} :

$$d_{libre} \geq z_{min} + 2 \tag{1.77}$$

Les coefficients de la fonction d'énumération de poids IRWEF moyenne $A^{C_P}(W, Z)$ d'un code PCC classique composé de 2 codeurs RSC identiques et d'un entrelaceur s'exprime comme suit [2] :

$$A^{C_P}(w, Z) = \frac{[A^C(w, Z)]^2}{\binom{K}{w}} \tag{1.78}$$

L'utilisation de la borne par réunion permet alors d'obtenir une borne supérieure sur le taux d'erreurs bit du décodeur à maximum de vraisemblance sur un canal à bruit blanc additif gaussien (BBAG).

On définit la distance libre effective $d_{libre,eff}$ du code concaténé comme suit :

$$d_{libre,eff} = 2 + 2z_{min} \tag{1.79}$$

Afin de maximiser z_{min} et donc $d_{libre,eff}$ il faut choisir un polynôme primitif au dénominateur. Pour un nombre de cellule M et un rendement R fixés, il reste alors à déterminer le ou les polynômes numérateurs. Dans [7], une recherche exhaustive des meilleurs codes constituants RSC pour les codes PCC et MPCC basée sur le critère de la distance libre effective a été effectuée.

Sur la figure (1.26), nous présentons les performances moyennes $P_e = f(E_b/N_0)$ d'un codeur convolutif concaténé en parallèle de rendement total $R_t = 1/3$ composé de 2 codeurs convolutifs récursifs systématiques RSC(7,5)et RSC(15,17) et pour 2 tailles d'entrelaceurs uniformes (100 et 1000bits) en utilisant la borne par union. On peut en particulier vérifier le gain d'entrelacement égal ici à $1/K$.

[2] $\binom{n}{p}$ est le nombre de combinaisons sans répétition de n éléments pris p à p. $\binom{n}{p} = \frac{n!}{p!(n-p)!}$.

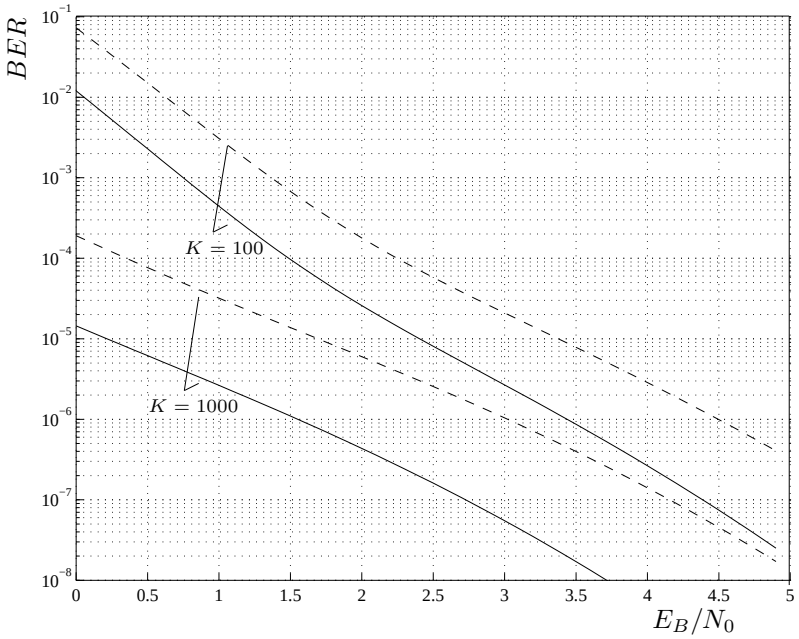


FIG. 1.26 – Comparaison des performances moyennes de codes PCC de rendement $R_t = 1/3$ composé de codes RSC(7,5) (en trait pointillé) et RSC(15,17) (en trait continu) et d'un entrelaceur uniforme $K = 100$ et $K = 1000$

1.7.3 Décodeur itératif pour les turbo codes

Dans le chapitre précédent nous avons montré que l'application de l'algorithme Somme-Produit sur un graphe sans cycle permet de calculer efficacement et exactement les probabilités *a posteriori* $APP(x_i^S = a)$. Cependant dans de nombreuses applications comme les codes concaténés (codes LDPC, codes produits, codes PCC,..), le graphe contient des cycles. Malgré l'absence de justification mathématique rigoureuse, il est néanmoins possible d'utiliser l'algorithme Somme-Produit en faisant abstraction des cycles présents dans le graphe. C'est ce que nous allons faire en décodant itérativement les codes concaténés pour estimer les bits émis.

La complexité du calcul exact de $APP(x_i^S)$ dans une structure concaténée croit exponentionellement avec la longueur de la séquence d'information. L'intérêt principal de la structure concaténée du codeur réside dans la possibilité de réaliser un décodage sous optimal de ce code. L'objectif du décodage itératif est d'exploiter la structure du graphe du codeur PCC composé de 2 graphes élémentaires sans cycle pour factoriser la probabilité *a posteriori* $APP(x_i^S = a) = Pr\{x_i^S = a \mid \mathbf{y^S}, \mathbf{y^1}, \mathbf{y^2}\}$. On évite ainsi de calculer la probabilité *a posteriori* de façon globale.

Le décodage initialement proposé par Berrou *et als.* dans [9] consiste à décoder alternativement les codes constituants du codeur concaténé. Pour chaque code élémentaire on effectue un décodage à entrées et sorties pondérées.

Le schéma simplifié du décodeur itératif est donné figure 1.27.

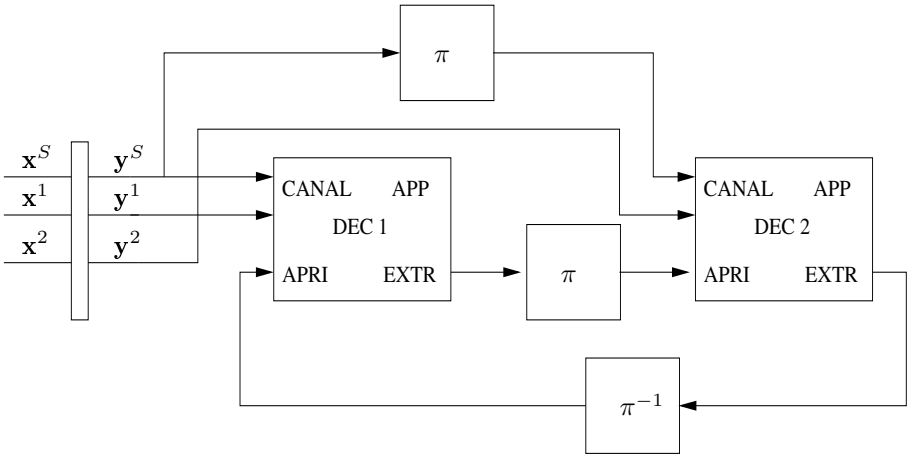


FIG. 1.27 – Structure du décodeur itératif

Sur cette figure, les différents étages de retard ne sont pas représentés mais doivent être implantés lors d'une réalisation matérielle pour tenir compte des retards des différents éléments du décodeur itératif (décodeur, entrelaceur).

Pendant le décodage, les informations extrinsèques calculées par le décodeur 1 après entrelacement deviennent les informations *a priori* pour le décodeur 2. Suite au décodage du second code, les informations extrinsèques du décodeur 2, après application de la fonction d'entrelacement inverse, sont transmises au décodeur 1 comme informations *a priori*. Ce processus appelé itération est répété plusieurs fois. Un exemple de graphe factorisé pour un code PCC de rendement 1/3 est donné en figure 1.28.

Ce graphe comporte de nombreux cycles. Il est important dans le cas des tailles de bloc moyenne ($K \leq 2000$) de chercher à réduire leur influence afin d'améliorer les performances du décodage itératif. Dans [35] et [27] les auteurs ont montré que le décodage itératif proposé par Berrou est équivalent à l'application de l'algorithme Somme-Produit sur le graphe factorisé du code.

Nous allons maintenant décrire les différentes opérations réalisées au cours du décodage.

Soit $\mathbf{u} = \mathbf{x}^S$, la séquence d'information, $\mathbf{c}^1$ et $\mathbf{c}^2$ les séquences des bits de redondance et $\mathbf{y}^S$, $\mathbf{y}^1$ et $\mathbf{y}^2$ les séquences reçues associées. A partir des $N = Kn/k$ observations relatives au

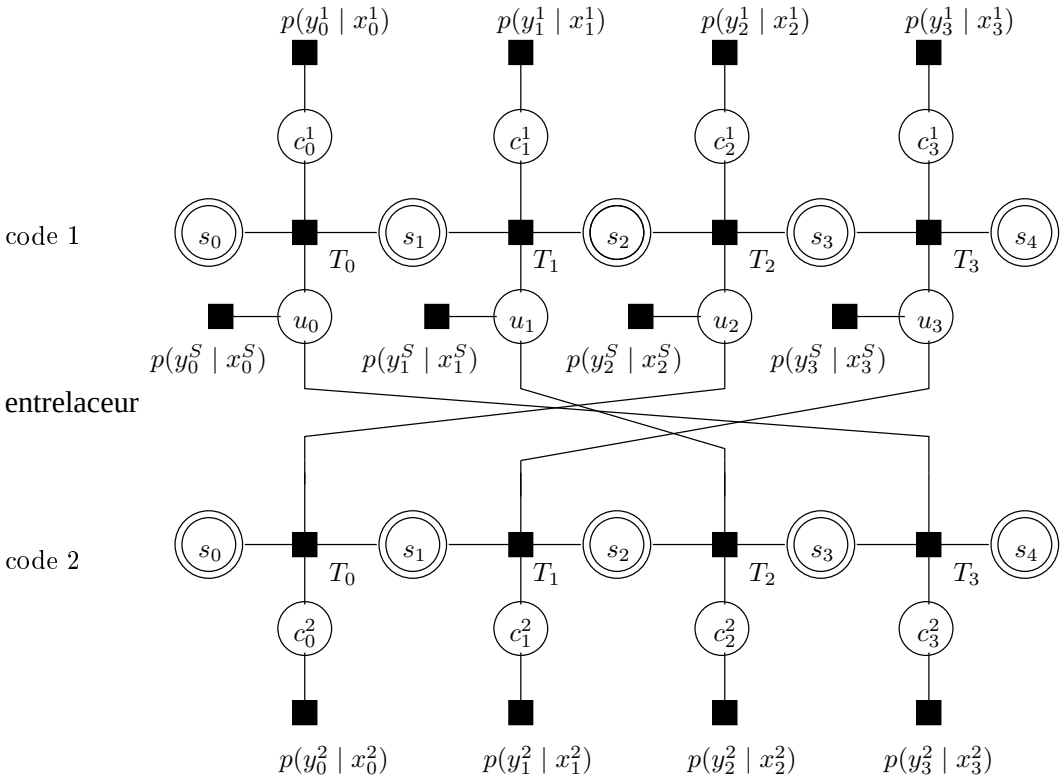


FIG. 1.28 – Exemple de graphe factorisé d'un turbo code de rendement 1/3

premier codeur et des probabilités *a priori* $APRI^{1(l)}(x_i^S)$ calculées lors d'un décodage précédent, le décodeur 1 calcule alors les probabilités extrinsèques $EXTR^{1(l)}(x_i^S)$ suivantes pour les K bits d'information et à chaque itération l, $1 \leq l \leq l_T$:

$$EXTR^{1(l)}(x_i^S = a) = \sum_{\mathbf{x^S}:x_i^S=a} \prod_{\substack{j=0 \\ j\neq i}}^{K-1} APRI^{1(l)}(x_j^S) p(y_j^S \mid x_j^S) \prod_{j'=0}^{N-K-1} p(y_{j'}^1 \mid x_{j'}^1) \tag{1.80}$$

$EXTR^{1(l)}(x_i^S = a)$ est la probabilité extrinsèque et $APRI^{1(l)}(x_i^S = a)$ est la probabilité *a priori* sur le i-ième bit de la séquence d'entrée $\mathbf{u}$. $APRI^{1(l)}(x_i^S = a)$ est obtenue comme suit :

$$APRI^{1(l)}(x_i^S) = \begin{cases} 1/2 \quad \forall i & \text{pour} \quad l = 1 \\ EXTR^{2(l-1)}(x_i^S) & \text{pour} \quad l > 1 \end{cases}$$

De la même façon, le décodeur 2 calcule $EXTR^{2(l)}(x_i^S)$:

$$EXTR^{2(l)}(x_i^S = a) = \sum_{\mathbf{x^S}:x_i^S=a} \prod_{\substack{j=0 \\ j\neq i}}^{K-1} APRI^{2(l)}(x_j^S) p(y_j^S \mid x_j^S) \prod_{j'=0}^{N-K-1} p(y_{j'}^2 \mid x_{j'}^2) \tag{1.81}$$

avec $APRI^{2(l)}(x_i^S) = EXTR^{1(l)}(x_i^S)$.

Finalement, pour réaliser la décision finale lors de la dernière itération l_T, on calcule $APP^{(l_T)}(x_i^S = a)$:

$$APP^{(l_T)}(x_i^S = a) \propto \sum_{\mathbf{x^S}:x_i^S=a} \prod_{j=0}^{K-1} APRI^{2(l_T)}(x_j^S) p(y_j^S \mid x_j^S) \prod_{j'=0}^{N-K-1} p(y_{j'}^2 \mid x_{j'}^2) \tag{1.82}$$

$$\propto p(y_i^S \mid x_i^S) \times APRI^{2(l_T)}(x_i^S = a) \times EXTR^{2(l_T)}(x_i^S = a) \tag{1.83}$$

Pour calculer ces probabilités, le décodeur applique l'algorithme Aller-Retour ou l'algorithme de Viterbi à sorties pondérées mais dans ce cas les performances obtenues seront sensiblement moins bonnes.

Quelques itérations suffisent généralement pour que le décodeur itératif converge. Le nombre d'itérations dépend cependant du rapport E_b/N_0. Au lieu de fixer à l'avance le nombre d'itérations, il est possible d'utiliser des critères d'arrêt comme l'écart entropique (*cross-entropy* en anglais) [36]. La prise de décision finale est réalisée à partir du signe des informations *a posteriori* obtenues lors de la dernière itération.

Comme pour les codes LDPC, au lieu de calculer les probabilités $EXTR(x_i^S = a)$ et $APP(x_i^S = a)$, on peut calculer les logarithmes des rapports de vraisemblance LLR (*logarithm likelihood ratio* en anglais) :

Ainsi, le logarithme du rapport de vraisemblance $L_{APP}(x_i^S)$ sera égal à :

$$L_{APP}(x_i^S) = L_{APRI}(x_i^S) + L_{INT}(x_i^S) + L_{EXTR}(x_i^S) \tag{1.84}$$

Avec

$$L_{INT}(x_i^S) = \ln \frac{p(y_j^S \mid x_j^S = 1)}{p(y_j^S \mid x_j^S = 0)} = \frac{2}{\sigma^2} y_j^S$$

$L_{INT}(x_i^S)$ est l'information intrinsèque ou information issue du canal. En pratique, il sera nécessaire d'estimer la variance du bruit σ^2.

1.7.4 Résultats de Simulation

Pour illustrer les performances remarquables des turbo codes nous présentons sur la figure 1.29 les courbes $BER = f(E_B/N_0)$ d'un turbo code de rendement $R_T = 1/2$ composé de 2 codes convolutifs RSC $(1, \frac{1+D^2}{1+D+D^2})$ et d'un entrelaceur optimisé *S-random* de dimension $K = 1024$. Une opération de perforation permet de passer d'un rendement de $1/3$ à $R_T = 1/2$. Les courbes sont données pour $l = 2, 5, 10$ et 15 itérations. On peut voir qu'au dela de 5 itérations le gain apporté par le décodage itératif est très faible (moins de 0.2 dB).

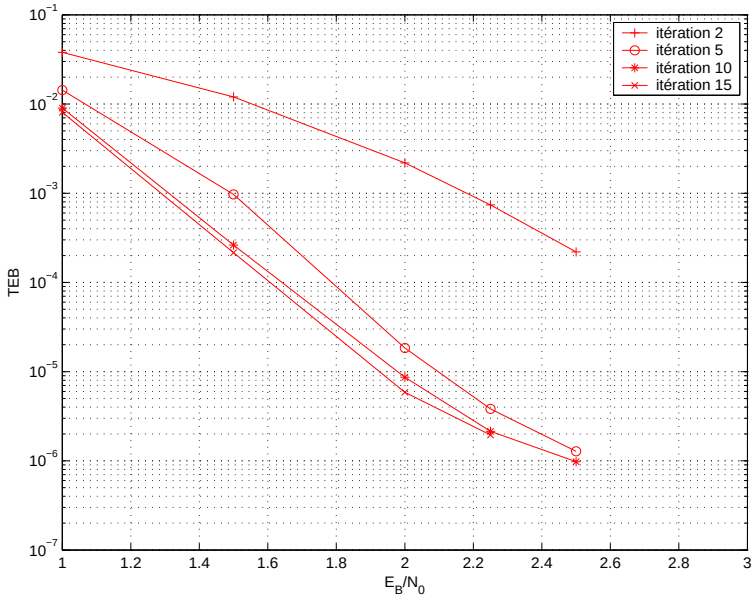


FIG. 1.29 – Courbes $BER = f(E_B/N_0)$ pour un turbo code de rendement $1/2$ composé de 2 codes convolutifs RSC $(1, \frac{1+D^2}{1+D+D^2})$ et d'un entrelaceur *S-random* de $K = 1024$.

Sur la figure 1.30 nous présentons les courbes $BER = f(E_B/N_0)$ d'un turbo code de rendement $R_T = 1/2$ composé de 2 codes convolutifs RSC $(1, \frac{1+D^4}{1+D+D^2+D^3+D^4})$ perforés et d'un entrelaceur pseudo aléatoire de dimension $K = 65536$. Ce code est le turbo code initialement proposé par Berrou and als en 1993 [9]. On peut voir qu'après 18 itérations, pour un BER de 10^{-5} on a $E_B/N_0 = 0.7$ dB. Nous sommes donc seulement à 0.7 dB de la limite de Shannon. Il faut cependant noter l'effet de palier (changement de pente de la courbe) qui apparait ici pour un BER de l'ordre de 10^{-6}. Cet effet de palier est caractéristique de la faible distance minimale des turbo codes. Cependant, l'optimisation de l'entrelaceur permet de le réduire sensiblement.

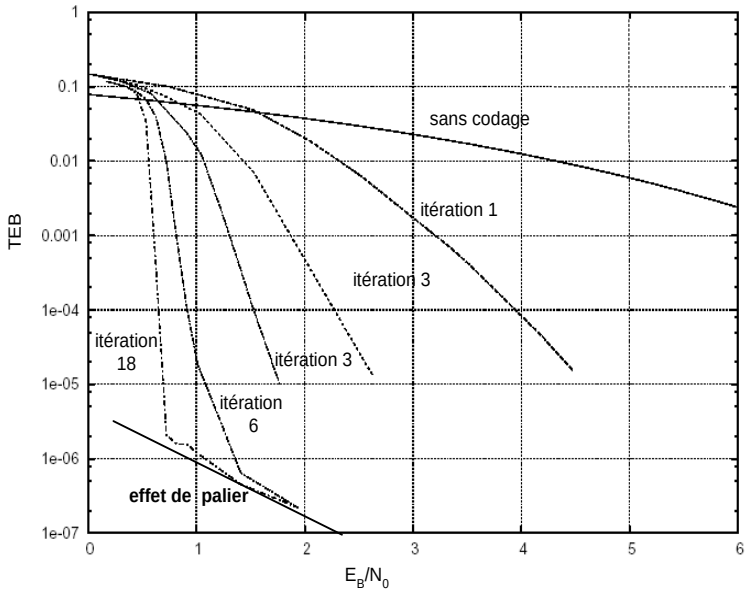


FIG. 1.30 – Courbes $BER = f(E_B/N_0)$ pour un turbo code de rendement $1/2$ composé de 2 codes convolutifs RSC $(1, \frac{1+D^4}{1+D+D^2+D^3+D^4})$ et de 1 entrelaceur pseudo aléatoire de dimension K=65536.

En conclusion, l'association turbo codes et décodage itératif permet d'obtenir des performances remarquables avec une complexité raisonnable.

1.8 Codes convolutifs concaténés en série

La structure du codeur associé aux codes convolutifs concaténés en série (SCC) (*serial concatenated convolutional* en anglais) est présentée figure 1.31. Ce codeur comprend 2 codeurs convolutifs systématiques C_1 et C_2 binaires respectivement de rendement k_1/n_1 et k_2/n_2 et 1 entrelaceur de Kn_1/k_1 bits. La longueur de la séquence d'entrée est limitée à K bits.

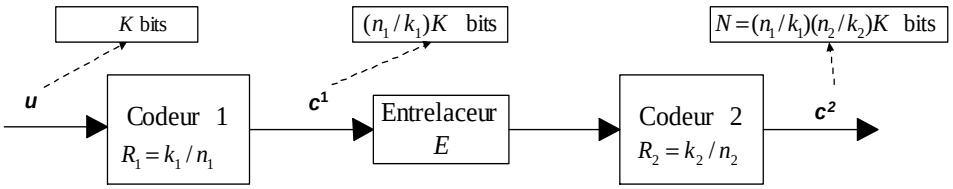


FIG. 1.31 – Codeurs convolutifs concaténés en série

Le code SCC ainsi construit est un code en bloc (N, K) avec $N = (\frac{n_2 n_1}{k_1 k_2})K$ en négligeant les éventuels bits de terminaison. Le rendement global du code est donc égal à :

$$R_T = \frac{k_1}{n_1}\frac{k_2}{n_2}$$

Par exemple en choisissant un code C_1 de rendement $1/2$ et un code C_2 de rendement $2/3$ on obtiendra un code SCC de rendement $R_T = 1/3$.

Comme pour les turbo codes, le décodage est itératif.

1.9 Codes RA

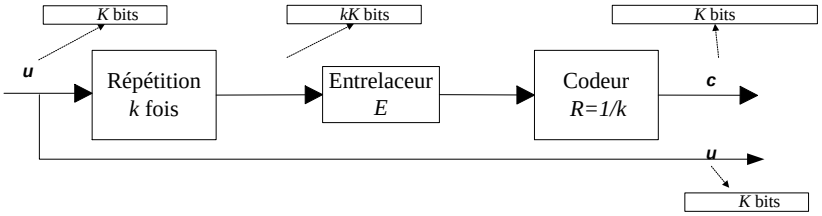


FIG. 1.32 – Codeur RA $R_T = 1/2$

Si on chaîne les codeurs constituants d'une structure de codeurs convolutifs concaténés en parallèle, alors le code ainsi obtenu peut être vu comme un code concaténé en série composé d'un code de répétition et d'un code convolutif. Si le codeur convolutif ne comprend qu'une mémoire interne on appelle ce codeur un accumulateur. Ce code est appelé code accumulation répétition RA (*repeat accumulate* en anglais).

1.10 Codes produits

Les codes produits ont été introduits en 1954 par Elias [15].
La structure de ces codes est donnée sur la figure 1.33.

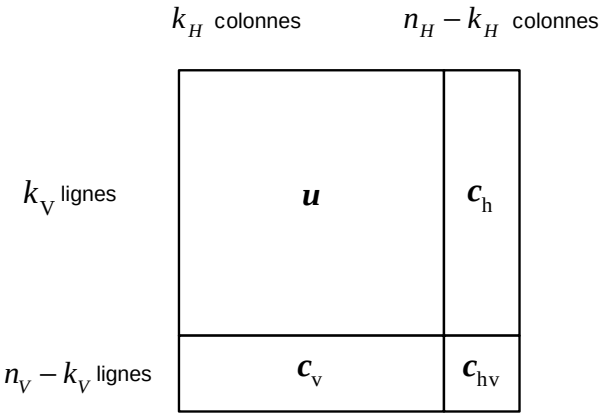


FIG. 1.33 – Code Produit.

Le code ainsi obtenu est un code linéaire en bloc systématique (N, K) avec $N = n_V n_H$ et $K = k_V k_H$.
Le rendement global du code produit est égal à :

$$R_T = \frac{k_V k_H}{n_V n_H}$$

Chacune des n_V lignes correspond à un code en bloc linéaire systématique C_H (n_H, k_H). Par ailleurs, chaque colonne correspond également à un code en bloc linéaire systématique C_V (n_V, k_V).
La distance minimale de ces codes est égale au produit $d_V \times d_H$ où d_V et d_H sont respectivement la distance minimale des codes constituants C_V et C_H. La distance minimale est supérieure à celle des codes PCBC grâce à la présence de bits de contrôle sur les bits de contrôle.
Comme pour les codes PCBC, les codes constituants sont principalement des codes de parité, de codes de Hamming ou des codes BCH.
Historiquement, les codes produits ont surtout été utilisés pour la protection des données des CD audio et des CD rom. Pour ces applications, les codes constituants sont des codes Reed-Solomon construits sur $GF(2^q)$ et le décodage est à décision dure et sans itération.
Cependant, pour s'approcher des performances du décodeur à maximum de vraisemblance, il est nécessaire de réaliser un décodage itératif.
Ces codes ont été recemment choisis pour la norme de réseau sans fil large bande IEEE 802.16. Les codes constituants sont des codes de parité (8,7),(16,15) et (32,31) ainsi que des codes de Hamming étendu (16,11), (32,26) et (64,57).

Chapitre 2

Applications des codes correcteurs d'erreurs

2.1 Communications spatiales

Dans toutes les communications spatiales la modulation utilisée est une modulation à 2 états de phase (BPSK).

• Mission Mariner en 1969 et Viking (Mars)
code en bloc Reed-Muller (32,6,16), décodeur à entrées pondérées

• Mission Pionner 9 en 1969 (soleil) Pionner 10 en 1972 (jupiter) Pionner 11 en 1973, Pionner 12 (vénus), Helios A et B (soleil)
code convolutif rendement $1/2$ $K_c = 32$, décodeur séquentiel à entrées pondérées

• Mission Voyager 1 et 2 en 1977 (jupiter et saturne) et standard CCSDS
code convolutif (163,171) rendement $1/2$, $K_c = 7$, décodeur de Viterbi à entrées pondérées

• Satellites Globalstar
code convolutif rendement $1/2$, $K_c = 9$

• Standard CCSDS et Mission Voyager
code convolutif (163,171) rendement $1/2$, $K_c = 7$, code Reed-Solomon (255,223,33)

• Mission Galileo en 1990 (jupiter)
code convolutif rendement $1/4$, $K_c = 15$, $d_{min} = 35$, the Big Viterbi Decoder (BVD)

• Mission Cassini (saturne)
code convolutif rendement $1/6$ $K_c = 15$, décodeur de Vitebi

2.2 Diffusion Audio et Télévision Numérique

• Télétext :
code de Hamming étendu (8,4)

• FM Digital Audio Broadcast (DAB) :
code cyclique difference set (273,191), décodage à logique majoritaire (entrées dures ou pondérées)

• Digital Audio Broadcast (DAB) pour le système audio DVB et MPEG :

code convolutif récursif rendement 1/2 $K_c = 5$, perforé (grand choix pour permettre une dégradation progressive des performances)

• système par satellite DirectTV, Digital Video Broadcast (DVB) satellite :
modulation QPSK + code convolutif (163,171) rendement 1/2 , $K_c = 7$, perforation 3/4,4/5, 5/6 et 7/8 , code Reed-Solomon (204,188,17)

• DVB terrestre :
modulation multiporteuses OFDM, QAM64 + code convolutif (163,171) rendement 1/2 , $K_c = 7$, perforation 3/4,4/5, 5/6 et 7/8 , code Reed-Solomon (204,188,17)

2.3 Transmission de données

• V29 en 1976 2400 b/s , half duplex :
modulation QAM16, 4b/symb

• V32 en 1984 9600 b/s full duplex :
modulation codée en treillis 32-CROSS 2 bits non codés et 2 bits codés avec code convolutif (3,2) $K_c = 4$ 4b/symb

• V33 en 1986 14400 b/s full duplex :
modulation codée en treillis 128-CROSS 4 bits non codés et 2 bits codés avec code convolutif (3,2) $K_c = 5$ 6b/symb

• V34 en 1994 33600 b/s full duplex :
modulation codée en treillis 4D, jusqu'à 1664 points, 8 bits non codés et 2 bits codés avec code convolutif (3,2) $K_c = 5$ 10b/symb

2.4 Stockage de données

• disque dur magnétique :
TBD

• CD audio :
RS(28,24) + RS(32,28) soit un rendement R=3/4

• CD ROM :
code produit (1170,1032) composé de codes lignes RS(26,24) et de codes colonnes RS(45,43)dans $GF(2^8)$

2.5 Radio communications

• GSM :
modulation GMSK, code convolutif rendement 1/2, $K_c = 5$

2.6 Transfert de fichiers

• TCP/IP :
CRC 16 bits sur l'entête, ARQ
• HDLC :
CRC 16 bits du CCITT ou CRC 32 bits optionnel, ARQ

Annexe 2
Calcul de la fonction IRWEF d'un code RSC

Cette méthode de calcul de la fonction d'énumération de poids IRWEF a été proposée par Viterbi *et al.* [49]. Nous allons présenter cette méthode à partir d'un exemple. Considérons le cas du codeur convolutif récursif systématique RSC(7,5). La matrice de transition d'état pour ce code est la suivante :

$$C(W,Z) = \begin{matrix} 00 \\ 00 \\ 10 \\ 11 \end{matrix} \begin{pmatrix} 1 & 0 & WZ & 0 \\ WZ & 0 & 1 & 0 \\ 0 & W & 0 & Z \\ 0 & Z & 0 & W \end{pmatrix} \tag{2.1}$$

et la fonction d'énumération de poids s'écrit :

$$A(W,Z) = (1 \quad 0 \quad 0 \quad 0) C^K(W,Z) \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.2}$$

Les équations de transitions sont alors :

$$\begin{pmatrix} A^{00} & (W,Z,i) \\ A^{01} & (W,Z,i) \\ A^{10} & (W,Z,i) \\ A^{11} & (W,Z,i) \end{pmatrix} = \begin{pmatrix} 1 & 0 & WZ & 0 \\ WZ & 0 & 1 & 0 \\ 0 & W & 0 & Z \\ 0 & Z & 0 & W \end{pmatrix} \begin{pmatrix} A^{00} & (W,Z,i-1) \\ A^{01} & (W,Z,i-1) \\ A^{10} & (W,Z,i-1) \\ A^{11} & (W,Z,i-1) \end{pmatrix}$$

et

$$\begin{cases} A^{(00)}_{w,z}(i) = A^{(00)}_{w,z}(i-1) + A^{(10)}_{w-1,z-1}(i-1) \\ A^{(01)}_{w,z}(i) = A^{(00)}_{w-1,z-1}(i-1) + A^{(10)}_{w,z}(i-1) \\ A^{(10)}_{w,z}(i) = A^{(01)}_{w-1,z}(i-1) + A^{(11)}_{w,z-1}(i-1) \\ A^{(11)}_{w,z}(i) = A^{(01)}_{w,z-1}(i-1) + A^{(11)}_{w-1,z}(i-1) \end{cases} \tag{2.3}$$

avec les conditions initiales suivantes : $A^{(00)}_{0,0}(0) = 1$, et $A^{(s)}_{w,z}(0) = 0$ sinon. Il suffit donc d'appliquer les équations 2.3 pour $i = 1, \ldots, K$.

Bibliographie

[1] L. R. Bahl, J. Cocke, F. Jelinek, J. Raviv. *"Optimal decoding of linear codes for minimizing symbol error rate"*. IEEE Trans. Inform. Theory **20**, pp. 284–287, Mars 1974.

[2] G. Battail, M. C. Decouvelaere, P. Godlewski. *'Replication decoding"*. IEEE Trans. Inform. Theory **25**(2), pp332–345, Mai 1979.

[3] G. Battail, C. Berrou, A. Glavieux. *"Pseudo-random recursive convolutional coding for near-capacity performance"*. Proc. of GLOBECOM 1993, Houston, Texas, USA, pp. 23–27, 1993.

[4] G. Battail. *"Théorie de l'information"*. Edition Masson, 1997.

[5] S. Benedetto, G. Montorsi. *"Design of parallel concatenated convolutional codes"*. IEEE Trans. on Comm. **44**(5), pp. 591–600, Mai 1996.

[6] S. Benedetto, G. Montorsi. *"Unveiling Turbo codes : Some results on parallel concatenated coding schemes"*. IEEE Trans. Inform. Theory **42**(2), pp. 409–428, 1996.

[7] S. Benedetto, R. Garello, G. Montorsi. *"A search for good convolutional codes to be used in the construction of Turbo codes"*. IEEE Trans. on Comm. **46**(9), pp. 1101–1105, Sept. 1998.

[8] E. R. Berlekamp. *"Algebraic coding theory"*. Edition Mc Graw-Hill,New York, 1968.

[9] C. Berrou, A. Glavieux, P. Thitimajshima. *"Near Shannon limit error correcting coding and decoding : Turbo-codes"*. Proc. of the 1993 Int. Conf. on Comm., Geneva, Switzeland, pp. 1064–1070, Mai. 1993.

[10] R.C Bose, D. K. Ray-Chaudhuri *"On a class of error correcting binary group codes"*. Inform. Control, vol.3, pp 68–79, mars 1960.

[11] J.B. Cain, G. C. Clark, J. M. Geist *"Punctured convolutional codes of rate $n-1/n$ and simplified maximum likelihood decoding"*. IEEE Trans. Inform. Theory **25**, pp. 97–100, Janvier 1979.

[12] D. Chase. *"A class of algorithms for decoding block codes with channel measurement information"*. IEEE Trans. on Inform. Theory. **18**, pp. 170–182, Jan. 1972.

[13] G. Cohen *"Codes correcteurs d'erreurs"*. Edition Masson, 1990.

[14] T. M. Cover et J. A. Thomas. *"Elements of information theory"*. Edition Wiley, 1991.

[15] P. Elias. *"Error-free coding"*. IRE Trans. Inform. Theory, pp. 29–37, 1954.

[16] G. D. Forney. *"Concatenated codes"*. MIT Press, Cambridge,Mass., 1966.

[17] G. D. Forney. *"Convolutional codes I : Algebraic structure"*. IEEE Trans. Inform. Theory **16**(6), pp. 720–738, 1970.

[18] G. D. Forney. *"The Viterbi algorithm"*. Proceedings of the IEEE **61**(3), pp. 268–278, Mars 1973.

[19] R. G. Gallager. *"Low density parity-check codes"*. MIT Press, Cambridge,Mass., 1963.Trans. Inform. Theory **25**, pp. 97–100, Janvier 1979.

[20] R. G. Gallager. *"Information theory and reliable communication"*. Edition Wiley, 1968.

[21] P. Guinand, J. Lodge. *"Trellis termination for Turbo encoders"*. 17th Biennal Symp. on Communications, Kingston, Canada, pp. 389–392, Mai 1994.

[22] J. Hagenauer, E. Offer, L. Papke. *"Iterative decoding of binary block and convolutional codes"*. IEEE Trans. Inform. Theory **42**(2), pp.429–445, Mar. 1996.

[23] A. Hocquenghem. *"Codes correcteurs d'erreurs"*. Chiffres, vol.2, pp 147–156, sept. 1959.

[24] J. Hokfelt, O. Edfors, T. Maseng. *"A survey on trellis termination alternatives for Turbo codes"*. Proc. of Vehicular Technology Conference, Houston, USA, pp. 2225–2229, Mai 1999.

[25] D. A. Huffman. *"A method for the construction of minimum redundancy codes"*. Proc. IRE, vol. 40, pp. 1098–1101, septembre 1952.

[26] R. Johannesson, K. S. Zigangirov. *"Fundamentals of convolutionnal coding"*. IEEE Press, Piscataway, NJ, USA, 1999.

[27] F. R. Kschischang, B. J. Frey, H. A. Loeliger. *"Factor graphs and the sum-product algorithm"*. IEEE Trans. Inform. Theory **47**(2), pp. 498–519, Feb. 2001.

[28] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman. *"Improved low density parity codes using irregular graphs and belief propagation"*. in Proc. of the 1998 Int. Symp. on Info. Theory,Boston, USA, pp.117, 1998..

[29] D. J. C. MacKay. *"Good error-correcting codes based on very sparse matrices"*. IEEE Trans. Inform. Theory **45**, pp. 399–431, Mar. 1999.

[30] R. J. McEliece, E. R. Rodemich, H. C. Rumsey, L. R. Welch. *"New upper bounds on the rate of a code via the Delsarte-MacWilliam"*. Proc. IEEE Trans. on Info. theory, vol. IT-23, pp.157–166, 1977.

[31] D. J. C. MacKay. *"Good error-correcting codes based on very sparse matrices"*. IEEE Trans. Inform. Theory **45**, pp. 399–431, Mar. 1999.

[32] D. J. MacKay. *"Information Theory, Inference and Learning Algorithms"*. Téléchargeable sur le site htt ://www.inference.phy.cam.ac.uk/mackay

[33] F. J. MacWilliams, N. J. A. Sloane *The theory of error correcting codes* North Holland Elsevier, Amsterdam, The Netherlands, 1977

[34] J. L. Massey *"Shift register synthesis and BCH decoding"*. IEEE Trans. Inform. Theory **6**, pp. 459–470, Janvier 1960.

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

[36] M. Moher. *"Decoding via cross entropy minimization"*. Proc. of GLOBECOM 1993, Houston, USA, pp 809–813, nov. 1993.

[37] W. W. Peterson, E. J. Weldon *Error correcting codes* MIT Press, Cambridge, MA, 1972

[38] J. G. Proakis *"Digital communications"*. McGraw-Hill,Boson,USA, 4^{eme} édition, 2001.

[39] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq. *"Near optimum product codes"*. Proc. of GLOBECOM 1994, San Francisco, USA, pp. 339–343, 1994.

[40] I. S. Reed, G. Solomon. *"Polynomial codes over certain finite fields"*. Journal SIAM, vol.8, pp 300–304, 1960.

[41] T. Richardson, R. Urbanke. *"The capacity of low density parity check codes under message passing algorithm"*. IEEE Trans. Inform. Theory **47**(2), pp. 599–617, Feb. 2001.

[42] T. Richardson, A. Shokrollahi, R. Urbanke. *"Design of capacity approaching irregular low density parity check codes"*. IEEE Trans. Inform. Theory **47**(2), pp. 619–637, Feb. 2001.

[43] J. J. Rissanen. *"Generalized Kraft inequality and arithmetic coding"*. IBM Journal Research and Dev., vol. 20, No. 3,pp. 198–203, mai 1976.

[44] C. Shannon. *"A mathematical theory of communication"*. Bell Syst. Tech. J., vol. 27, pp. 623–659 et pp. 623–656, juillet et octobre 1948. Téléchargeable sur le site http ://www.math.psu.edu/gunesh/Entropy/shannon.ps ou http ://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

[45] B. Sklar. *"A primer on Turbo Code concepts"*. IEEE Communications Magazine , pp. 94–102, Dec. 1997.

[46] M. Van Dijk, S. Egner, R. Motwani, A. Koppelaer. *"Simultaneous zero-tailing of parallel convolutional codes"*. Proc. of the 2000 Int. Symp. on Info. Theory, Sorrento, Italia, pp. 368, Juin 2000.

[47] A. J. Viterbi. *"Error bounds for convolutional codes and an asymptotically optimum decoding algorithm"*. IEEE Trans. Inform. Theory **13**, pp. 260–269, Avril 1967.

[48] A. J. Viterbi, J. K. Omura. *"Principes des communications numériques"*. Edition Dunod, CNET-ENST, 1982.

[49] A. M. Viterbi, A. J. Viterbi, J. Nicolas, N. T. Sindhushayana *"Perspectives on interleaved concatenated codes with iterative soft output decoding"*. Proc. of the Int. Symp. on Turbo Codes and Related Topics, Brest, France, Sept. 1997.

[50] N. Wiberg, H. A. Loeliger, R Kotter. *"Codes and iterative decoding on general graphs"*. European Trans. on Telecomm.,**6**(5), pp. 513–526, Sept. 1995.

[51] Y. Yasuda, K. Kashiki, H. Hirata. *"High rate punctured convolutionnal codes for soft-decision Viterbi decoding"*. IEEE trans. on Communications **32**, pp. 315–319, Mars 1984.

[52] J. Ziv et A. Lempel. *"Compression of individual sequences via variable rate coding"*. Proc. IEEE Trans. on Info. theory, vol. IT-24, No. 5,pp. 530–536, septembre 1978.

Table des matières