ÉCOLE DOCTORALE Science - Technologie - Santé

Année : 2003 **Thèse N°**

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline / Spécialité : Électronique

présentée et soutenue par

Emmanuel Cadic

le Vendredi 24 Octobre 2003

Construction de Turbo Codes courts possédant de bonnes propriétés de distance minimale

Thèse dirigée par Thierry Berger et Raymond Quéré

Jury :

Président

Jean-Michel Nébus	Professeur à l'Université de Limoges

Rapporteurs :

Pascale Charpin	Directeur de Recherches à l'INRIA-Rocquencourt
Claude Berrou	Directeur d'Études, École Nationale Supérieure des Télécommunications, Brest
Gilles Zémor	École Nationale Supérieure des Télécommunications, Paris (HDR)

Examinateurs :

Raymond Quéré	Professeur de l'Université de Limoges, co-directeur
Thierry Berger	Professeur de l'Université de Limoges, co-directeur
Jean-Claude Carlach	Chercheur France Télécom R&D, co-directeur
Jean-Pierre Tillich	Maître de Conférences de l'Université Paris XI

Remerciements

Je souhaite remercier Jean-Michel Nébus d'avoir accepté de présider mon jury de thèse ainsi que Pascale Charpin, Claude Berrou et Gilles Zémor qui ont bien voulu en être les rapporteurs. Leurs remarques judicieuses ont grandement contribué à l'amélioration de ce document final.

Je remercie également Dominique Lacroix-Penther pour m'avoir accueilli durant ces 3 années dans son équipe du laboratoire DMR/DDH de France Télécom R&D à Rennes. Je souhaite exprimer toute ma sympathie à mes collègues et amis thésards, ex-thésards ou permanents pour nos discussions scientifiques, culturelles mais aussi pour les bons moments passés ensembles.

Je tiens à exprimer toute ma gratitude et mon amitié à Jean-Claude Carlach qui m'a permis de développer mon goût pour la recherche et qui m'a soutenu durant ces 3 années. Par ses encouragements, son enthousiasme et sa bonne humeur, il m'a permis de mener cette thèse jusqu'à son terme.

Merci également à Thierry Berger et Jean-Pierre Tillich pour leur disponibilité et l'aide qu'ils ont su m'apporter. Je remercie aussi les thésards du projet "Cortex", Ayoub Otmani et Gregory Olocco, pour les échanges que nous avons pu avoir.

Merci à ma famille et à mes amis pour avoir été près de moi durant ces 3 ans.

Table des matières

Table des figures

Liste des tableaux

Résumé

L'objectif de cette thèse est de réaliser des turbo codes possédant de bonnes distances minimales et de contribuer ainsi à repousser le phénomène "d'error floor" qui correspond à un seuil de l'ordre de 10^{-6} pour le taux d'erreur résiduelles binaires en dessous duquel la pente de la courbe de TEB diminue de façon significative. Ce problème s'est sensiblement amélioré avec l'apparition des codes duo-binaires de Berrou [11] qui permettent notamment d'obtenir de meilleures distances minimales.
Pour obtenir de bonnes distances minimales avec des turbo codes courts (longueur inférieure à 512), la construction initialement utilisée et étudiée dans cette thèse a été celle proposée par Carlach et Vervoux [26] qui permet d'obtenir d'excellentes distances minimales mais qui malheureusement s'avère moins performante en terme de décodage notamment pour des raisons propres à la structure. Après avoir identifié les raisons qui empêchent un décodage efficace de cette famille de codes, nous faisons évoluer ces codes en utilisant des structures graphiques différentes reposant toujours sur l'assemblage de codes composants de petite complexité. L'idée est de réaliser ce changement sans pour autant perdre les qualités de distance minimale de ces codes et par conséquent il est nécessaire de comprendre pourquoi les distances minimales de cette famille initiale de codes sont bonnes et de définir un critère de choix pour les codes composants. Le critère de choix ne dépend pas de la distance minimale des codes composants mais du polynôme de transition de ces codes et permet donc de sélectionner des codes composants de très faible complexité qui sont assemblés de façon à générer des treillis cycliques à seulement 4 états. Ces treillis sont alors utilisés pour élaborer des turbo codes parallèle ou série présentant de bonnes distances minimales. Certains codes auto-duaux extrémaux sont notamment construits ainsi.

Abstract

This thesis is aimed at building turbo codes whith good minimum distances and delaying the "error-floor" which corespond to a threshold of 10^{-6} for the binary error rate. Under this threshold, the slope of the curve decreases significantly. This problem is alleviated by the use of duo-binary turbo codes [11] which guarantee better minimum distances.
In order to obtain good minimum distances with short turbo codes (length inferior to 512), the first construction used and studied is the one proposed by Carlach and Vervoux [26]. It allows to obtain very good minimum distances but its decoding is unfortunately very difficult because of its structure. After identifying the reasons for this problem, we have modified these codes by using some graphicals structures which are the gathering of low complexity components codes. The idea is to realize this change without loosing the minimum distances properties, and consequently we had to understand why minimum distances are good for this familly of codes and define a new criteria to choose "good" components codes. This criteria is independent from the minimum distance of the component codes because it is derived from the Input-Output Weight Enumerator ($IOWE$) of the components codes. It allows us to choose components codes with very low complexity which are combined in order to provide 4-state tail-biting trellises. These trellises are then used to build multiple parallel concatenated and serial turbo codes with good minimum distances. Some extremal self-dual codes have been built in that way.

Notations et abréviations

Notations

$\mathcal{C}(n,k,d)$	Code en bloc de longueur n, de dimension k et de distance minimale d
$\mathcal{C}^{\perp}(n,n-k,d^{\perp})$	Code en bloc dual de $\mathcal{C}(n,k,d)$
d_{min} ou d	Distance minimale d'un code en bloc
$d_E(x,y)$	Distance euclidienne entre x et y
$d_H(x,y)$	Distance de Hamming entre x et y
$\frac{E_b}{N_0}$	Rapport entre l'énergie binaire et la densité spectrale de puissance du bruit
$\mathbb{F}_q$	Corps fini à q éléments
$\mathcal{R}_{\mathcal{C}}$	Rendement $\frac{k}{n}$ du code $\mathcal{C}$
$(S_A\|S_B)$	Concaténation de deux sections de treillis S_A et S_B

Abréviations

APP	*A Posteriori* Probability
$AWGN$	Additive White Gaussian Noise
$BCJR$	Bahl, Cocke, Jelinek and Raviv Algorithm
$IOWEF$	Input-Output Weight Enumerating Function
$LDPC$	Low-Density Parity-Check (Codes)
LLR	Log-Likelihood Ratio
MAP	Maximum *a posteriori*
MAQ-M	Modulation d'Amplitude en Quadrature à M états
MCB	Matrice Circulante par Bloc
MCT	Modulation Codée en Treillis
MDP-M	Modulation de Phase à M états
RA	Repeat-Accumulate
$SISO$	Soft-In/Soft-Out
$SOVA$	Soft-Out Viterbi Algorithm
TEB	Taux d'Erreur Binaire

Introduction

La première communication hertzienne réalisée en 1897 à Londres permit à son inventeur Marconi de se voir décerner un Prix Nobel de physique en 1909. Jusqu'en 1945, la totalité des transmissions, civiles ou militaires, sont analogiques et avaient déjà une grande importance commerciale (radio-diffusion, communications militaires, ...). L'aspect numérique était alors uniquement utilisé pour crypter les communications militaires ou diplomatiques. Ce n'est qu'à la fin des années 1950 que l'on voit apparaître les premiers systèmes civils de radiotéléphonie mobile et encore ceux-ci sont réservés au service public et à quelques professionnels aux États-Unis. Il faudra attendre la fin des années 1970 pour que la France soit dotée de la première génération de radiotéléphonie cellulaire dont la diffusion restera relativement confidentielle. Cependant les premières communications numériques rendues publiques proviennent de la NASA avec les missions vers les planètes Mars, Jupiter (Mariner en 1969, Vicking) ou sondes spatiales (Pioneer début des années 1970). Le véritable essor des communications radio-mobiles provient du début des années 90 avec la deuxième génération de systèmes mobiles appelée GSM (Global System for Mobiles). Les applications radio sont multiples (diffusion hertzienne, télévision, téléphonie mobile, transmission par satellite, ...) et la quantité d'information qui y transite ne cesse d'augmenter avec la diversité d'offre des opérateurs (audio, vidéo, images,...) et bien évidemment le nombre d'utilisateurs toujours plus important. Le nombre d'abonnés en France aux services de téléphones mobiles atteint 38,5 millions fin 2002 et est encore en expansion de 4,3% par rapport à 2001. Avec un chiffre d'affaire de 3,1 milliards d'euros au quatrième trimestre 2002, en hausse de 17,5% par rapport au quatrième trimestre 2001, la crise du secteur des télécommunications est donc toute relative ! C'est l'un des rares secteurs économiques en expansion et d'une importance croissante pour les autres secteurs. Il est donc primordial pour France Télécom de rester à la pointe de la recherche et du développement de services et de technologies innovants notamment pour la couche physique de transmission.

En regard de ces chiffres, on comprend mieux la nécessité d'améliorer sans cesse la qualité des systèmes de communication numérique. Ce problème peut être abordé de différentes façons si l'on considère que sa qualité est le plus souvent évaluée en déterminant la probabilité d'erreur des symboles transmis. La première approche, sachant que cette probabilité est dépendante du rapport signal à bruit, consiste à augmenter ce rapport par exemple en ayant une puissance de signal plus importante à l'émission. En plus de son coût très important, cette méthode s'avère bien souvent impossible à mettre en application le plus souvent pour des raisons techniques comme ce peut être le cas avec les équipements embarqués qui disposent d'une énergie limitée. Une seconde approche, plus complexe à mettre en place, consiste à introduire des techniques de codage pour les messages à transmettre. Cette dernière solution est aujourd'hui la plus retenue et c'est notamment elle qui a motivé la réalisation de cette thèse. Elle s'inscrit dans le domaine de la théorie des codes correcteurs d'erreurs dont la naissance date de la fin des années 40 avec l'explosion des télécommunications. Il est impossible d'évoquer ce phénomène sans mentionner l'article fondateur de Shannon [83]. Il est ainsi parvenu à démontrer que tout système pouvait être conçu de façon à combattre efficacement les altérations du message engendrées par la présence de bruit sur le canal. Malheureusement la réponse n'y est pas explicitée et toutes les tentatives sont longtemps restées infructueuses, les bornes théoriques demeurant très éloignées des résultats obtenus en pratique. Hamming et ses travaux sur les codes correcteurs de 1945 ont également contribué à l'éclosion de cette nouvelle discipline appelée théorie de l'information qui traite de tous les problèmes théoriques ou pratiques permettant de transmettre ou de stocker de l'information. Les principaux jalons ont été les travaux d'Elias (codes produits, codes convolutifs) en 1954, de Reed et Muller (codes de Reed-Muller en 1954), de Bose, Chaudhury et Hockenghem (codes BCH en 1959), de Golay en 1959 (codes de Golay) ou de Reed et Solomon en 1960. En 1963, Gallager [48] publie sa thèse sur les codes $LDPC$ (Low Density Parity Check) qui vont être oubliés pendant 30 ans malgré leurs performances et l'importance théorique de ces travaux car les calculateurs numériques de l'époque n'étaient pas assez rapides et trop coûteux. Les travaux de Nordström et Robinson en 1967 ont connu aussi une longue période d'oubli car leur petit code optimal était apparemment non linéaire et très "exotique". On le redécouvre au début des années 90 avec l'engouement pour les codes sur $\mathbb{Z}_4$ [56] utiles pour la construction de modulations codées. En 1966 on voit apparaître les premiers codes concaténés avec les travaux de Forney. En 1969, Viterbi [94] applique une technique de programmation dynamique au calcul de bornes théoriques de taux d'er-

reurs et de vitesse de convergence au décodage des codes convolutifs sur un treillis. C'est de nouveau Forney [40] en 1970 qui soulignera l'intérêt pratique de cet algorithme pour décoder en temps réel un code convolutif. Cet algorithme est appelé "algorithme de Viterbi à décision dure"(0 ou 1). En 1974, Bahl, Cocke, Jelinek et Raviv publient l'algorithme dit maintenant $BCJR$ [6] permettant de calculer de façon optimale les probabilités *a posteriori* des bits transmis à partir du treillis du code convolutif et de leurs probabilités *a priori*. Cet algorithme se rapproche d'un algorithme de type Viterbi à décision douce (contrairement à la version "dure", les sorties ne sont plus binaires mais des valeurs réelles). En France, à la fin des années 80, Battail est le premier à s'intéresser au décodage à décision douce en proposant un algorithme soft dérivé du $SOVA$ utilisant ces probabilités ou leur rapport de vraisemblance ou le logarithme de ces rapports de vraisemblance, et surtout il a souligné, comme Wolf [101] ou l'artcile $BCJR$ [6], le fait que l'on pouvait considérer les codes en blocs comme des codes convolutifs en utilisant des représentations en treillis. En 1991, Berrou et Glavieux inventent les turbo codes qui vont révolutionner le domaine des codes correcteurs d'erreurs et amener un intérêt nouveau pour les études théoriques en théorie de l'information, notamment en combinant les principes de concaténation avec les algorithmes de décodage à décision douce. Cette invention a provoqué une explosion d'articles et un renouveau du domaine de la théorie de l'information en permettant de redécouvrir certaines familles de codes ($LDPC$) ou d'en découvrir de nouvelles (Woven-codes, Repeat Accumulate codes,...). Ces familles de codes puisent fréquemment leur intérêt dans leur faible complexité due à l'utilisation de treillis à très peu d'états par section.
L'utilisation de décodeurs souples est elle aussi un facteur qualitatif important, car par exemple, sur canal gaussien, ces décodeurs permettent de gagner près de 2 dB par rapport à un décodeur à décision dure. De plus ils sont très faciles à mettre en œuvre pour tous les codes acceptant des représentations en arbre (algorithmes de type max-plus ou somme produit) ou en treillis (algorithmes de type Viterbi ou $BCJR$). Cependant les résultats ainsi obtenus demeurent très en deçà des bornes théoriques si on les utilise avec les codes classiques antérieurs aux turbo codes. C'est là que les travaux de Berrou et al. [13] prennent tout leur intérêt en permettant, grâce à leur nouveau système de codage-décodage, d'obtenir des performances se rapprochant des bornes théoriques et atteignant une qualité de décodage en réelle progression notamment sur les canaux gaussiens. L'encodage des turbo codes initiaux est réalisé à l'aide de deux codes convolutifs (récursifs de préférence) concaténés parallèlement *via* un entrelaceur comme présenté sur la figure 1 (X représente l'information et Y_1 et Y_2 les redondances four-

nies par les deux codes convolutifs récursifs). Le décodage est réalisé à l'aide d'une variante de l'algorithme $BCJR$ qui permet d'extraire de l'information sur chacun des codes et de l'échanger entre ces deux codes. L'information extraite d'un code est dite "extrinsèque" et est réinjectée à l'itération suivante dans un autre code composant et surtout pas dans le même code composant afin de bénéficier de la diversité de codage. Le processus pouvant se répéter plusieurs fois, on parle de décodage itératif. L'appellation "turbo" provient justement de la réinjection des extrinsèques pour rappeler l'image des moteurs turbo qui réutilisent une partie de l'énergie gaspillée par le moteur pour donner plus de puissance au moteur. La première particularité marquante de ces turbo codes est que leurs courbes de TEB de décodage deviennent rapidement très pentues à faible rapport signal à bruit ($Eb/N0$) après quelques itérations.

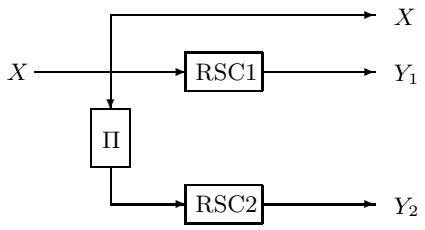


FIG. 1 – Turbo code parallèle de rendement $\frac{1}{3}$

Le second aspect marquant de ces turbo codes est qu'ils atteignent très rapidement un seuil de l'ordre de 10^{-6} pour le taux d'erreurs résiduelles binaires en dessous duquel la pente de la courbe de TEB diminue. Ce phénomène dit "d'error floor" est dû en partie à la faible distance minimale et à la distribution des poids des premiers turbo codes. Ce problème est bien amélioré avec les turbo codes duo-binaires de Berrou [11]. Cependant, ces codes ont enfin permis d'approcher à quelques dixièmes de dB des limites théoriques de Shannon pour des tailles de bloc de l'ordre de 64000. Mieux comprendre le fonctionnement des turbo codes est donc apparu comme un axe de recherche prioritaire. L'un des principaux éléments de réponse provient de la généralisation des graphes de Tanner [87] qui a permis de représenter de nombreuses familles de codes (codes produits, $LDPC$, turbo codes) sous des formes se prêtant bien à la description du décodage itératif. Bien qu'il existe plusieurs types d'algorithmes pour ces graphes de Tanner qui convergent vers la solution optimale lorsque les graphes sont dépourvus de cycles, tous présentent l'inconvénient de ne plus fonctionner dès que des cycles courts apparaissent sur la structure. Ce problème est

rédhibitoire car les turbo codes possèdent de nombreux cycles courts dans leurs représentations en graphes de Tanner, mais le plus handicapant demeure le fait que les codes présentant de bonnes distances minimales ne peuvent pas se représenter en graphe de Tanner sans faire apparaître des cycles courts [35].
La question finalement est d'améliorer les turbo codes notamment en repoussant le fameux phénomène "d'error floor" qui est en grande partie la conséquence d'une distance minimale trop faible. Les travaux de cette thèse s'inscrivent dans cette optique et prennent comme idée initiale la nouvelle construction inventée en 1998 par Carlach et Vervoux [26]. Cette construction utilise tout comme les turbo codes des permutations, et des codes très courts que l'on assemble entre eux de différentes façons pour obtenir des codes de rendement $\frac{1}{2}$ et de longueurs importantes. Des travaux ont été menés sur ces codes notamment par Olocco et Tillich [73][74] et Otmani [76][77]. Le but de cette thèse n'était bien evidemment pas de reprendre ces mêmes travaux théoriques mais de prendre dans un premier temps une voie plus axée sur le décodage et d'utiliser la faible complexité de décodage des petits codes de base pour réaliser le décodage global. Dans un second temps, les travaux se sont essentiellement orientés vers les critères de choix des petits codes composants et les méthodes d'assemblage pour obtenir des turbo codes possédant des distance minimales plus importantes.

Ce mémoire de thèse est constitué de cinq chapitres organisés comme suit :

- ◇ Le premier chapitre est consacré à quelques généralités sur les techniques de codage et les différentes méthodes de représentation des codes correcteurs d'erreurs. Nous introduisons aussi les deux grandes "familles" de codes (codes en blocs et codes convolutifs).

- ◇ Le second chapitre aborde la partie décodage en présentant différentes méthodes suivant que l'on dispose ou non de structures graphiques adaptées à la représentation du code. Le décodage sur les arbres ainsi que le décodage analogique sont dans la continuité de mon stage de DEA [16].

- ◇ Le troisième chapitre traite d'une famille de turbo codes appelé codes "cortex". Après quelques rappels sur la construction et ses propriétés déjà en grande partie étudiées, nous abordons le problème de leur décodage. Enfin une construction dérivée de ces codes, appelée codes

à information répartie, est présentée.

◇ Le quatrième chapitre présente une construction de turbo codes provenant de la concaténation parallèle de plusieurs treillis de faible complexité (4 états). Les motivations du choix de tels treillis résident dans l'étude des critères de transition de poids sur un treillis. Des résultats de décodage y sont également présentés.

◇ Le dernier chapitre est consacré à une généralisation des codes "cortex" *via* l'utilisation des treillis présentés dans le chapitre précédent. Les codes ainsi construit utilise donc aussi des treillis cycliques à 4 états et permettent d'obtenir des codes auto-duaux.

Chapitre 1

Techniques de codage

Pour être en mesure d'utiliser des outils mathématiques efficaces, on s'impose une contrainte de linéarité sur les codes correcteurs étudiés. Les codes les plus "puissants" ne sont pas nécessairement linéaires, mais les méthodes existantes pour rechercher les bons codes non linéaires sont rares et souvent difficiles à mettre en œuvre. De plus il existe de très bons codes linéaires et la restriction au cas linéaire n'est donc pas une contrainte trop importante.
Cependant il existe une méthode simple de recherche de codes non linéaires sur $\mathbb{Z}_2$. On recherche pour cela un code linéaire sur $\mathbb{Z}_4$ que l'on "projette" sur $\mathbb{Z}_2$ *via* le mapping de Gray. La distance minimale du code ainsi obtenu est alors la même que celle du code linéaire sur $\mathbb{Z}_4$ en raison de l'isométrie induite par le mapping utilisé entre les deux anneaux.
Dans ce chapitre nous nous proposons de présenter les deux grandes familles de codes linéaires que sont les codes en blocs et les codes convolutifs. Enfin nous rappelons quelques notions élémentaires sur les treillis qui sont des structures graphiques particulièrement bien adaptées à la représentation des codes mais aussi des modulations codées.

1.1 Codes en blocs

Cette famille de code très utilisée est particulièrement bien introduite par MacWilliams et Sloane [69].

1.1.1 Définitions

Nous considérons dans ce qui suit l'alphabet $\mathbb{F}_p$ où p est une puissance d'un nombre premier q ($p = q^r$) et les codes construits seront donc constitués par les éléments de ce corps.
Le codage en bloc consiste à associer à chaque bloc de k éléments d'information un bloc de n éléments ($n > k$) de façon à ajouter de la redondance pour une meilleure protection contre le bruit. Ces blocs de n éléments sont au nombre de 2^k et sont appelés les mots du code. L'ensemble de ces mots de code forment un sous-espace vectoriel de dimension k de l'ensemble $\mathbb{F}_p^n$ qui est constitué par tous les n-uplets possibles. De manière immédiate, la somme de deux mots de code est donc encore un mot de code, et le mot nul (toutes les composantes à 0) est toujours un élément du code. Un codage en bloc linéaire est par conséquent une application linéaire Φ de la forme :

$$\Phi : \left\{ \begin{array}{ccc} \mathbb{F}_p^k & \longrightarrow & \mathbb{F}_p^n \\ m & \longmapsto & c = \Phi(m) \end{array} \right.$$

où $m = (m_0, m_1, \ldots, m_{k-1})$ représente les k symboles d'information et $c = (c_0, c_1, \ldots, c_{n-1})$ le mot de code qui lui est associé.
Soient $\mathcal{E} = (e_0, e_1, \ldots, e_{k-1})$ et $\mathcal{F} = (f_0, f_1, \ldots, f_{n-1})$ des bases respectivement pour $\mathbb{F}_p^k$ et $\mathbb{F}_p^n$. Nous pouvons écrire la décomposition de m dans la base $\mathcal{E}$:

$$m = \sum_{i=0}^{k-1} m_i e_i$$

L'application Φ étant linéaire, nous avons :

$$c = \Phi(m) = \sum_{i=0}^{k-1} m_i \Phi(e_i)$$

Chacun des vecteurs $\Phi(e_i)$ peut s'exprimer dans la base $\mathcal{F}$ comme suit :

$$\Phi(e_i) = \sum_{j=0}^{n-1} \phi_{i,j} f_j$$

Les k vecteurs $\Phi(e_i) = (\phi_{i,0}, \phi_{i,1}, \ldots, \phi_{i,n-1})$ $(0 \preceq i \preceq k-1)$ composent les lignes de la matrice G associée à l'application linéaire Φ.

$$G = \begin{pmatrix} \phi_{0,0} & \cdots & \phi_{0,j} & \cdots & \phi_{0,n-1} \\ \vdots & & \vdots & & \vdots \\ \phi_{i,0} & \cdots & \phi_{i,j} & \cdots & \phi_{i,n-1} \\ \vdots & & \vdots & & \vdots \\ \phi_{k-1,0} & \cdots & \phi_{k-1,j} & \cdots & \phi_{k-1,n-1} \end{pmatrix}$$

Cette matrice G à k lignes et n colonnes est appelée matrice génératrice du code et permet d'associer au bloc d'information m un mot de code c par la relation $c = mG$. La matrice génératrice d'un code en bloc n'est pas unique et en particulier il est toujours possible de l'écrire, à permutation près, sous la forme suivante :

$$G = [Id_k, P] = \begin{pmatrix} 1 & 0 & \ldots & 0 & p_{0,1} & \cdots & p_{0,i} & \cdots & p_{0,n-k} \\ 0 & 1 & \ldots & 0 & p_{1,1} & \cdots & p_{1,i} & \cdots & p_{1,n-k} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & 1 & p_{k-1,1} & \cdots & p_{k-1,i} & \cdots & p_{k-1,n-k} \end{pmatrix}$$

où Id_k est la matrice identité $k \times k$ et P une matrice $k \times (n-k)$ permettant de calculer les $(n-k)$ éléments binaires de redondance. Les mots du code s'exprime sous la forme $c = (m, mP)$ et le code est alors dit systématique.

1.1.2 Dualité

Définition 1.1.1 *Soient $x = (x_0, x_1, \ldots, x_{n-1})$ et $y = (y_0, y_1, \ldots, y_{n-1})$. Ces deux vecteurs de $\mathbb{F}_{q^r}$ sont othogonaux si leur produit scalaire est nul, c'est-à-dire si*

$$x.y = \sum_{i=0}^{n-1} x_i y_i \equiv 0 \mod q$$

À tout code en blocs linéaire $\mathcal{C}(n,k)$ on peut associer un code dual $\mathcal{C}^{\perp}(n, n-k)$. Dans ce cas tout mot de $\mathcal{C}^{\perp}(n, n-k)$ est orthogonal à tout mot de $\mathcal{C}(n,k)$. Le code dual $\mathcal{C}^{\perp}(n, n-k)$ est donc un sous-espace vectoriel de $\mathbb{F}_p^n$, de dimension $(n-k)$ constitué de q^{n-k} mots de longueur n. Soient H la matrice génératrice de $\mathcal{C}^{\perp}(n, n-k)$ et c un mot de code quelconque de $\mathcal{C}(n,k)$, alors :

$$cH^T = 0$$

Définition 1.1.2 *Soit y un vecteur de $\mathbb{F}_p^n$. y est un mot du code $\mathcal{C}(n,k)$ si et seulement si il est orthogonal aux mots de $\mathcal{C}^\perp(n,n-k)$, c'est-à-dire si :*

$$yH^T = 0$$

La matrice H est appelée matrice de contrôle de parité du code $\mathcal{C}(n,k)$. Cette matrice se détermine simplement dans le cas où la matrice génératrice du code considéré est systématique. En effet, comme $c = mG$, alors $GH^T = 0$. De plus la matrice génératrice étant systématique, alors $G = [Id_k, P]$ et donc

$$H = [-P^T, Id_{n-k}]$$

Exemple
Soit le code de Hamming [7,4,3] de matrice génératrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

sa matrice de contrôle est alors

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

1.1.3 Distance minimale

Nous considérons dans cette partie uniquement les codes linéaires binaires et nous introduisons dans un premier temps la notion de distance de Hamming.

Définition 1.1.3 *On appelle distance de Hamming $d(x,y)$ entre deux mots x et y de $\mathbb{F}_2^n$ le nombre de composantes pour lesquelles ils diffèrent.*

Définition 1.1.4 *Le poids de Hamming d'un mot x est sa distance au mot nul : $w(x) = d(x,0)$.*

Exemple

$$\begin{array}{rcl} x = (1110100) & \Longrightarrow & w(x) = 4 \\ y = (0101100) & \Longrightarrow & w(y) = 3 \\ d(x,y) = w(x-y) & = 1+1+1 = 3 & \end{array}$$

Définition 1.1.5 *On appelle distance minimale d'un code $\mathcal{C}$, notée d_{min}, la plus petite distance de Hamming entre deux mots distincts du code. Autrement dit,*

$$d_{min} = \min_{\substack{c_i, c_j \in \mathcal{C} \\ c_i \neq c_j}} d(c_i, c_j)$$

Définition 1.1.6 *La distance minimale d'un code $\mathcal{C}$, notée d_{min}, est égale au poids minimal de ses mots non nuls :*

$$d_{min} = \min_{\substack{c_i \in \mathcal{C} \\ c_i \neq 0}} w(c_i)$$

L'évaluation de la distance minimale d'un code en blocs est un problème très complexe. Pour des codes de petite dimension il suffit cependant d'énumérer de façon exhaustive les mots de code et de déterminer celui de poids le plus faible. Il est également possible de la déterminer à l'aide de la matrice de contrôle de parité H car la distance minimale d_{min} est égale au plus petit nombre de colonnes linéairement dépendantes de cette matrice. Malheureusement dans la plupart des cas, les calculs étant trop complexes à mener, il faut se résoudre à déterminer uniquement une borne supérieure de cette distance minimale.

Théorème 1.1.1 *(Borne de Singleton) Soit $\mathcal{C}$ un code de paramètres $[n, k, d_{min}]$, alors :*

$$d_{min} \preceq n - k + 1$$

Théorème 1.1.2 *(Borne de Plotkin simplifiée) Soit $\mathcal{C}$ un code de paramètres $[n, k, d_{min}]$, alors :*

$$d_{min} \preceq \frac{n 2^{k-1}}{2^k - 1}$$

Cette notion de distance minimale est d'autant plus importante qu'elle possède un lien direct avec le pouvoir de détection et de correction d'un code en blocs.

Théorème 1.1.3 *Soit $\mathcal{C}$ un code de paramètres $[n, k, d_{min}]$. Il est capable de détecter toutes les configurations de $(d_{min} - 1)$ erreurs dans un bloc de n éléments binaires et il permet de corriger toute erreur de poids au plus $t = \lfloor \frac{d-1}{2} \rfloor$. La valeur t s'appelle la capacité de correction du code.*

La distance minimale intervient aussi dans le cas des effacements où lorsqu'un symbole est douteux on préfère l'effacer.

Proposition 1.1.1 *Soit $\mathcal{C}$ un code de distance minimale d. Tout algorithme décodage de C de capacité de correction t peut-être transformé en algorithme de correction d'erreurs et deffacements de capacité t' erreurs et f effacements avec $2t' + f \preceq 2t$.*

1.2 Codes convolutifs

Cette famille de codes est sans doute actuellement la plus couramment rencontrée, notamment en raison de leur utilisation pour l'élaboration des turbo codes. Une très bonne introduction à ces codes est par exemple fourni dans [59].

1.2.1 Principe

Les codes convolutifs constituent l'une des principales famille de codes correcteurs d'erreurs. Dans un code convolutif, chaque bloc de n éléments en sortie du décodeur dépend non seulement du bloc composé des k éléments positionnés à l'entrée du décodeur mais aussi des m blocs précédents. Cette famille de codes fait donc appel à un effet de mémoire d'ordre m et la quantité $(m + 1)$ s'appelle la longueur de contrainte K du code. De même que pour les codes en blocs, la quantité $R = \frac{k}{n}$ s'appelle le rendement du code, et si les k éléments d'information présents à l'entrée du décodeur sont effectivement transmis (c'est-à-dire apparaissent explicitement dans le bloc de n éléments), le code est dit systématique.

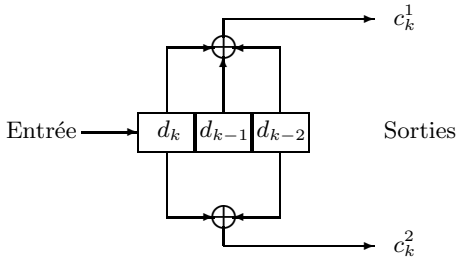


FIG. 1.1 – Exemple de codeur convolutif non systématique

La figure 1.1 présente l'exemple d'un codeur convolutif de rendement $R = \frac{1}{2}$ et de longueur de contrainte $(m + 1) = 3$. Ce codeur accepte en entrée des blocs de $k = 1$ élément binaire et en sortie des blocs de $n = 2$ éléments binaires.

Chacune des sorties du codeur est égale au produit de convolution (d'où l'expression "convolutif") entre la suite binaire présente à l'entrée du codeur et la réponse de ce même codeur définie par ses séquences génératrices. Pour l'exemple de la figure 1.1, ces séquences sont données par :

$$\forall i \in \{0,1\},\ g_i = [g_{i0}, g_{i1}, g_{i2}]$$

avec

$$\forall i \in \{0,1\},\ c_k^i = \sum_{j=0}^{2} g_{ij} d_{k-j} \text{ et } g_{ij} \in \{0,1\}$$

Les séquences génératrices sont fréquemment exprimées en octal, ce qui dans le cas de la figure 1.1 donne

$$\begin{cases} g_1 &= [1,1,1] = 7_{(octal)} \\ g_2 &= [1,0,1] = 5_{(octal)} \end{cases}$$

Remarque 1.2.1 *Les sorties du codeur étant égales à des combinaisons linéaires des éléments d'information présent à l'entrée du codeur convolutif, le code est nécessairement linéaire.*

On peut également définir les codes convolutifs avec une approche plus algébrique en considérant leurs polynômes générateurs qui s'expriment en fonction d'une variable D (pour "Delay"). Toujours dans le cadre de notre exemple, les polynômes générateurs ont pour forme générale

$$\begin{cases} G_1(D) &= g_{10} + g_{11}D + g_{12}D^2 \\ G_2(D) &= g_{20} + g_{21}D + g_{22}D^2 \end{cases}$$

ou encore

$$\begin{cases} G_1(D) &= 1 + D + D^2 \\ G_2(D) &= 1 + D^2 \end{cases}$$

1.2.2 Représentations

La représentation des codes convolutifs par les polynômes ou les matrices n'est pas réellement adaptée au décodage. Aussi pour réaliser les décodeurs, on utilise des représentations graphiques de ces codes de type diagramme en arbre, diagramme d'état ou treillis. La représentation en treillis est très efficace pour le décodage car elle est parfaitement adaptée aux algorithmes de type $BCJR$ ou $SOVA$ et est notamment utilisée par les turbo codes. Dans cette partie nous donnons des exemples de ces 3 différents types de représentations qui sont ici toutes les trois associées au codeur de la figure 1.1.

1.2.2.1 Diagramme en arbre

La figure 1.2 représente un tel diagramme en arbre et utilise deux conventions :

- ◇ Le temps s'écoule de la gauche vers la droite
- ◇ Lorsque l'élément à l'entrée du codeur est 0 (resp. 1), le couple binaire en sortie de codeur (noté entre parenthèses) étiquette une branche montante (resp. descendante). Ces branches sont liées en des points appelés nœuds. Pour une séquence d'information fournie à l'entrée du codeur, la séquence associée en sortie du codeur correspond à un chemin de l'arbre qui est constitué d'une succession de branches.

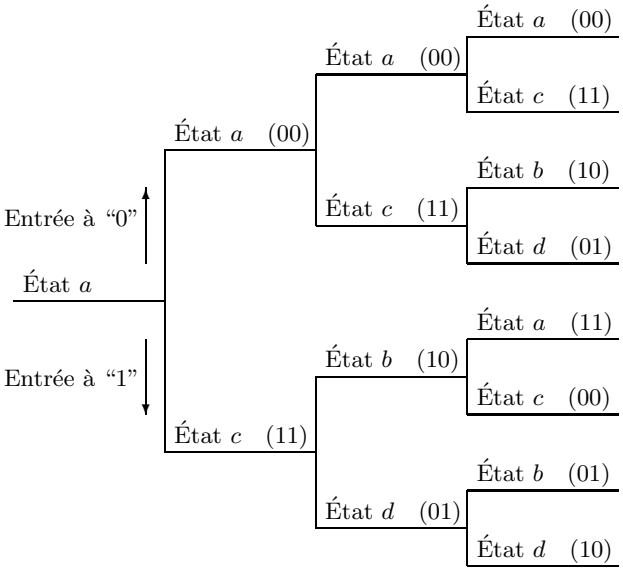


FIG. 1.2 – Diagramme en arbre du codeur convolutif

1.2.2.2 Diagramme en treillis

Cette représentation correspond à la façon dont communiquent les états entre eux en fonction du temps. Sur la figure 1.3, les liens (branches) entre états représentés en pointillés et en traits pleins correpondent respectivement à la présence d'un bit d'information égale à 0 et 1. Chacune de ces branches fournit un couple de bits pour la sortie du codeur.

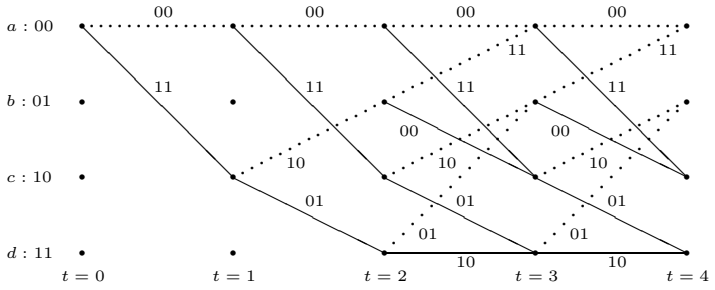


FIG. 1.3 – Diagramme en treillis du codeur convolutif

Remarque 1.2.2 *Ces représentations en treillis sont périodiques et après* $(m+1)$ *décalages, le motif du treillis se répète.*

1.2.2.3 Diagramme d'état

La figure 1.4 représente un tel diagramme et se singularise par rapport aux deux représentations précédentes par le fait qu'elle n'utilise pas un axe temporel de façon explicite. Elle ne présente que les différents états du codeur et leurs liens.

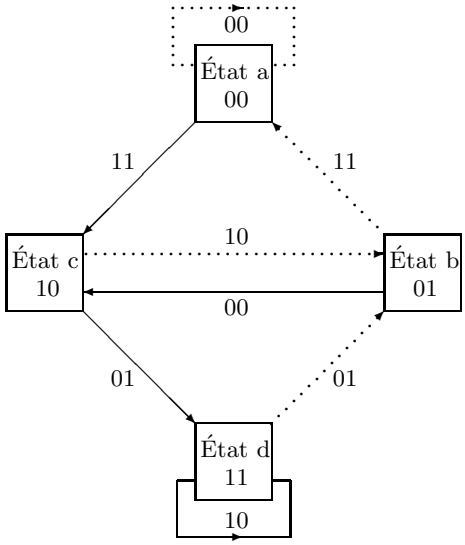


FIG. 1.4 – Diagramme d'état du codeur convolutif

1.2.3 Notion de distance

La distance sert toujours ici, comme pour les codes en blocs, à caractériser la capacité de correction, mais sa signification dépend également des algorithmes de décodage mis en œuvre. Pour les codes convolutifs, on définit généralement deux types de distance :

◇ La distance minimale :

$$d_{min} = \min_{m \in \mathcal{M}} w(m)$$

où $\mathcal{M}$ est l'ensemble des mots non nuls de dimension finie k.
Cette distance concerne plus particulièrement l'étude des propriétés de correction des décodeurs algébriques pour lesquels le décodage s'effectue sur une observation limitée de n bits.

◇ La distance libre :

$$d_{free} = \min_{m \in \mathcal{M}_\infty} w(m)$$

où $\mathcal{M}_\infty$ est l'ensemble des mots non nuls de dimension non limitée.
Cette distance concerne plus particulièrement l'étude des décodeurs probabilistes pour lesquels l'observation utilisée au décodage n'est pas limitée à n bits.

Remarque 1.2.3 *Ces deux distances vérifient l'inégalité :*

$$d_{min} \preceq d_{free}$$

1.3 Généralités sur les treillis

La recherche du treillis minimal d'un code est un problème complexe comme l'ont montré Köetter et al. [63] mais très important car le choix du treillis va déterminer en grande partie la complexité du décodage. Dans cette partie nous faisons simplement une introduction sur les notions de base concernant les treillis et leur minimalité.

1.3.1 Rappels

Un graphe orienté labélisé est un triplet $(\mathcal{S}, \mathcal{B}, \mathcal{A})$ où $\mathcal{S}$ est un ensemble de sommets, $\mathcal{B}$ un ensemble de branches et $\mathcal{A}$ un alphabet. Un diagramme en treillis $\mathcal{T}$ est un graphe orienté labélisé avec la propriété que $\mathcal{S}$ peut se diviser en $n+1$ ensembles disjoints $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup ... \cup \mathcal{S}_n$ de telle façon que si $(s, s', \alpha) \in \mathcal{T}$ alors $s \in \mathcal{S}_{i-1}$ et $s' \in \mathcal{S}_i$ pour $i \in \{1, 2, ..., n\}$. Le paramètre

n est appelé longueur de $\mathcal{T}$.
Un treillis conventionnel est un diagramme en treillis possédant un unique sommet (ou état) de départ et un unique état d'arrivée, c'est-à-dire $|\mathcal{S}_0| = |\mathcal{S}_n| = 1$. Un treillis cyclique est un diagramme en treillis pour lequel $|\mathcal{S}_0| = |\mathcal{S}_n|$. Chaque chemin de longueur n entre $\mathcal{S}_0$ et $\mathcal{S}_n$ est un vecteur de $\mathcal{A}^n$ pour lequel la $i^{\text{ième}}$ composante est l'étiquette de la branche reliant les états $\mathcal{S}_{i-1}$ et $\mathcal{S}_i$. Dans un treillis conventionnel, tout chemin d'un état initial de $\mathcal{S}_0$ vers un état final de $\mathcal{S}_n$ est un chemin valide, tandis que pour un treillis cyclique un chemin est valide si l'état de départ de $\mathcal{S}_0$ est identique à celui d'arrivée de $\mathcal{S}_n$.

Définition 1.3.1 *Soit $\mathcal{T} = (\mathcal{S}, \mathcal{B}, \mathbb{F}_q)$ un treillis de longueur n. On dit que $\mathcal{T}$ représente un code en bloc $\mathcal{C}$ de longueur n sur $\mathbb{F}_q$ si et seulement si l'ensemble des chemins valides de $\mathcal{T}$ est isomorphe à l'ensemble des mots de code de $\mathcal{C}$.*

On dit qu'un treillis $\mathcal{T}$ est plus petit ou égal à un autre treillis $\mathcal{T}'$ représentant le même code si $|\mathcal{S}_i| \preceq |\mathcal{S}'_i|$ pour tout $i \in \{0, 1, ..., n\}$. On note $\mathcal{T} \preceq_\Theta \mathcal{T}'$. Si $\mathcal{T}$ est strictement plus petit que $\mathcal{T}'$, on note $\mathcal{T} \prec_\Theta \mathcal{T}'$.

Définition 1.3.2 *On dit que $\mathcal{T}$ est minimal pour la relation $\prec_\Theta$, ou plus simplement minimal, s'il n'existe pas de treillis $\mathcal{T}'$ représentant le même code tel que $\mathcal{T}' \prec_\Theta \mathcal{T}$.*

Remarque 1.3.1 *Une difficulté majeure de la recherche des treillis minimaux d'un code provient du fait que le produit de treillis minimaux n'est pas nécessairement minimal.*

On associe la complexité d'un treillis, conventionnel ou cyclique, avec le nombre d'états qu'il comporte par section. La complexité d'état d'un treillis $\mathcal{T} = (\mathcal{S}, \mathcal{B}, \mathbb{F}_q)$ est la séquence $c(\mathcal{T}) = (c_0(\mathcal{T}), c_1(\mathcal{T}), ..., c_n(\mathcal{T}))$ où

$$c_i(\mathcal{T}) = \log_q(|\mathcal{S}_i|) \text{ pour } i \in \{1, 2, ..., n\}.$$

Cette séquence est aussi appelée "profil de complexité d'états" du treillis.
Il existe différentes bornes, inférieures et supérieures, concernant la complexité d'états. Ici, seule la borne supérieure la plus fréquemment utilisée est mentionnée.

Proposition 1.3.1 *Soit $\mathcal{T}$ un treillis minimal conventionnel associé à un code $\mathcal{C}$ de paramètres $[n, k, d]$. Alors,*

$$\max_{i \in \{0,1,...,n\}} c_i(\mathcal{T}) \preceq \min(k, n-k).$$

Proposition 1.3.2 *Soit $\mathcal{T}$ un treillis minimal cyclique associé à un code $\mathcal{C}$ de paramètres $[n, k, d]$. Alors,*

$$\max_{i\in\{0,1,\ldots,n\}} c_i(\mathcal{T}) \preceq \frac{1}{2} \times \min(k, n-k).$$

1.3.2 Exemples

Dans cette partie nous donnons quelques exemples de treillis associés au Hamming [8,4,4]. Le choix de ce code est double. Tout d'abord il intervient sous différentes formes dans les chapitres 3, 4 et 5 où il est utilisé comme code de base pour différentes construction. Ensuite ce code étant de petite dimension, il est simple à représenter. De plus ses représentations en treillis sont très différentes les unes des autres. La figure 1.5 représente un treillis cyclique ("tail-biting") régulier où chaque section comporte 4 états.

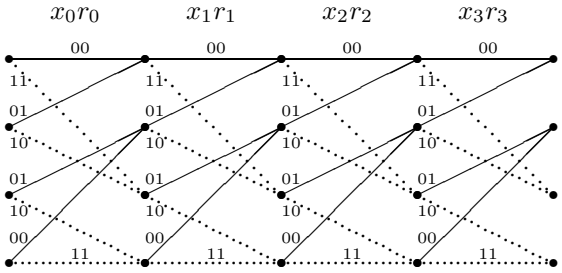


FIG. 1.5 – Treillis cyclique régulier du Hamming [8,4,4]

La figure 1.6 est bien distincte du treillis de la figure 1.5 puisqu'elle possède deux sections ne comportant qu'un seul états, les autres en possédant toutes 4. Ce treillis est donc irrégulier.

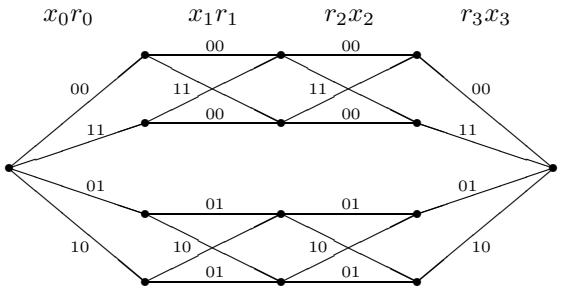


FIG. 1.6 – Treillis irrégulier du Hamming [8,4,4]

Enfin la figure 1.7 présente un cas encore différent puisque chaque section possède 2 états et le treillis est sous forme multi-branches (plusieurs branches distinctes entre deux états).

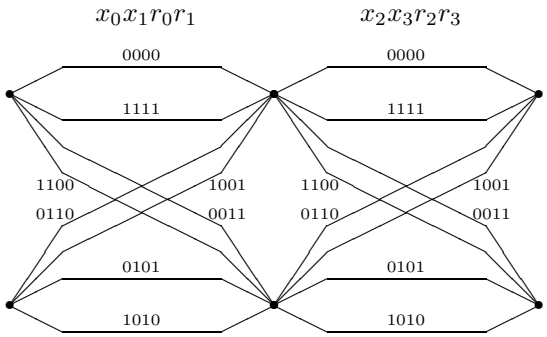


FIG. 1.7 – Treillis cyclique "multi-branches" du Hamming [8,4,4]

1.4 Modulation codées en treillis

Cette partie destinée à présenter une utilisation des treillis autres que sur des codes correcteurs classiques est également un prétexte pour faire quelques rappels élémentaires concernant notamment les modulations.
C'est en 1976 que la notion de MCT (ou TCM : Trellis Coded Modulation) a vu le jour avec l'introduction par Ungerboeck [89][90] du concept de codage en treillis dans l'espace des signaux. Les résultats établis à ce jour montrent qu'elles permettent, sur canal gaussien, d'obtenir des gains de codage variant de 3 à 6 dB par rapport aux modulations non codées de même efficacité spectrale et cela sans avoir recours à une réduction du taux effectif d'information portée par le message émis.
Les MCT peuvent être décrites de deux façons distinctes selon que l'on opte pour la description d'origine introduite par Ungerboeck ou la description analytique due à Calderbank et Mazo [22].

1.4.1 Quelques rappels sur les modulations

Afin d'assurer la transmission d'un message $m(t)$ entre un émetteur et un récepteur, il est nécessaire d'utiliser un signal intermédiaire, susceptible de se propager sous forme d'onde électro-magnétique dans le milieu de transmission, sur lequel le message $m(t)$ est imprimé par "modulation". Ce signal intermédiaire agit comme un support de transmission et peut s'écrire sous

la forme :

$$s(t) = A \times \cos(\omega_0 t + \phi_0)$$

où A, $\omega_0 (= 2\pi f_0)$ et ϕ_0 correspondent respectivement à l'amplitude, la fréquence et la phase du signal.
Le fait de faire varier l'un ou l'autre de ces trois paramètres donne lieu à une modulation du même nom à savoir modulation d' amplitude, de phase et de fréquence. Il est important de remarquer que dans tous les cas, la variation s'exprime en fonction du message à transmettre.
Dans le cas des premiers systèmes de communication, les fonctions de modulation et de codage correcteur d'erreurs étaient séparées, le modulateur et le démodulateur permettant de faire le lien entre le canal analogique de transmission et la source discrète traitée par le codeur et le décodeur.
Pour un système de modulation simple, durant chaque intervalle de modulation, le modulateur transforme m bits en l'un des $M = 2^m$ signaux transmissibles alors que le démodulateur décide quel est le signal de la constellation (ensemble des signaux associés à une modulation) le plus proche du signal reçu pour retrouver les m symboles binaires émis. La figure 1.8 présente quelques exemples de contellations parmi les plus classiques.

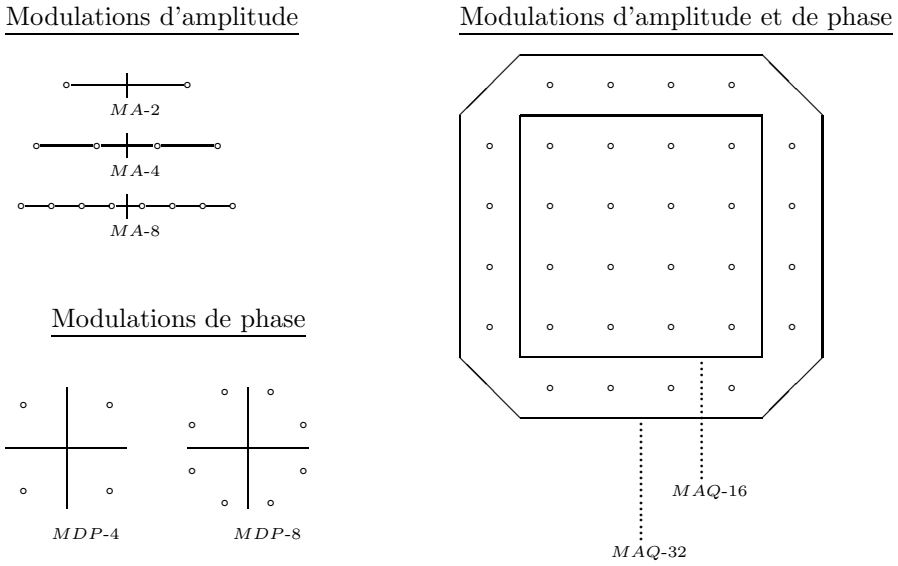


FIG. 1.8 – Exemples de constellations

De la même façon que la distance de Hamming, pour les codes correc-

teurs d'erreurs, est étroitement liée à la capacité de correction, la distance euclidienne est elle aussi en relation avec les capacités de correction d'un démodulateur. Elle est même le seul bon critère pour les modulations codées. En considérant un signal modulé comme un point d'un espace multidimensionnel, alors plus les points sont espacés et plus le nombre de "voisins" possibles pour le signal reçu est faible. Notons d^e_{min} la distance euclidienne minimale entre deux signaux d'une même constellation. Dans le cas d'une modulation numérique de phase (MDP), nous avons :

$$d^e_{min} = 2\sqrt{E}\sin\left(\frac{\pi}{M}\right)$$

où E est l'énergie utilisée pour transmettre un symbole.
Dans le cas d'une modulation d'amplitude en quadrature (MAQ), cette égalité devient :

$$d^e_{min} = A\sqrt{(2T)}$$

où A est l'amplitude et T l'inverse de la rapidité de modulation.
Ces quelques rappels concernent les modulations non-codées qui ont la particularité d'émettre des signaux non corrélés. Cependant, en présence de codage, les signaux émis par le modulateur ne sont plus indépendants. Il existe alors des séquences de signaux possibles que l'on nomme séquences codées et à l'inverse on parle de séquences interdites. Cette dépendance des séquences concerne également les MCT qui ont pour but d'augmenter la distance minimale entre les différentes séquences émises. Si l'on note d^c_{min} la distance minimale entre deux séquences codées, on peut déterminer le gain asymptotique de codage (ou gain à fort rapport signal à bruit), exprimé en décibels (dB), réalisé par la modulation codée à l'aide de la relation :

$$G_{dB} = 10\log\left(\frac{d^c_{min}}{d^e_{min}}\right)^2$$

1.4.2 Optimisation conjointe du codage et de la modulation

L'exemple présenté ici permet de mettre l'accent sur l'importance de concevoir simultanément le codage et la modulation, car lorsqu'ils sont réalisés de façon indépendante, les résultats obtenus peuvent être surprenant dans le sens où une modulation non-codée peut s'avérer plus performante à l'usage qu'un modulateur couplé à un codeur.
Considérons dans un premier temps une MDP-4 non-codée. Durant chaque intervalle de modulation, deux bits sont transmis. L'efficacité spectrale qui par définition correspond à la quantité de bits que l'on peut transmettre

à chaque modulation en fonction de la bande de fréquence disponible est donc de 2 $bit/s/Hz$. Dans ce cas, si l'on transmet sur un canal gaussien et que l'on estime le TEB pour un rapport signal à bruit (E_s/N_0) de 12.9 dB, on obtient environ 10^{-5}. Si l'on prend maintenant une MDP-8 non-codée,

Modulation utilisée	Rapport signal à bruit	Taux d'erreur binaire
MDP-4 non-codée	$12,9$ dB	10^{-5}
MDP-8 non-codée	$12,9$ dB	10^{-2}
MDP-8 codée	$12,9$ dB	10^{-5}

TAB. 1.1 – Comparaison de 3 modulations à rapport signal à bruit constant

toujours dans le cas d'un canal gaussien, pour un rapport signal à bruit de 12.9 dB, le taux d'erreur binaire est alors d'environ 10^{-2}. Cette différence s'explique par le fait que la constellation associée à la MDP-8 possède plus de points que pour la MDP-4 et que ceux-ci sont de surcroît à plus faible distance les uns des autres. On dit que la distance euclidienne minimale (distance minimale entre deux points d'une même constellation) de la MDP-4 est supérieure à celle de la MDP-8. Pour compenser cet écart de performance entre les deux modulations, on insère un code de rendement 2/3 et de distance minimale 7 entre la source et la modulation. Pour récupérer le message initialement émis, il est donc nécessaire, en sortie du canal, de décoder les symboles issus du démodulateur. Cela a pour conséquence immédiate d'augmenter la complexité du système, sans pour autant pouvoir la justifier par un véritable gain de performance.
Deux raisons majeures conduisent à cet état de fait :

- Tout d'abord, la décision dure et indépendante du décodeur qui est réalisée dans l'espace des signaux peut, lorsqu'elle est trop imprécise, entraîner une perte irréversible sur l'information reçue que le décodeur ne parviendra que très partiellement à compenser. Pour tenter de remedier à ce problème on peut s'orienter vers des techniques de décision optimales mais complexes. Par exemple, si l'on note $r_n = a_n + w_n$ la séquence bruitée reçue où a_n et w_n correspondent respectivement à la séquence codée émise et au bruit additif du canal de transmission, la règle de décision douce pour la séquence optimale s'établit de la façon suivante :
  Soit S l'ensemble des séquences de signaux codés que le codeur et le modulateur peuvent produire. La séquence de signaux a_n la plus

vraisemblablement émise vérifie :

$$|r_n - a_n|^2 = \min_{s_n \in S} |r_n - s_n|^2 = \min_{s_n \in S} \left(\sum_i |r_n^{(i)} - s_n^{(i)}|^2 \right)$$

Autrement dit, on recherche la séquence de S la plus proche de r_n pour le carré de la distance euclidienne.

- ◇ Le second problème qui apparaît provient de l'utilisation des métriques. En effet, si les codes sont conçus en terme de distance de Hamming, les modulations ne font appel qu'à des distances euclidiennes. Or, la transformation que subissent les symboles d'un code optimisé pour la distance de Hamming via une modulation ne garantit pas l'obtention d'une bonne distance euclidienne dans l'espace des signaux. En d'autres termes, on peut posséder un code doté d'une bonne distance minimale, et donc d'une capacité de correction acceptable, mais affaiblir ses performances s'il est couplé avec une modulation de distance minimale faible qui va induire une perte d'information parfois importante dès la sortie du démodulateur.

Ces deux phénomènes suffisent à mettre en avant l'intérêt qu'il peut exister de chercher à construire conjointement les codeurs et modulateurs.

1.4.3 Partitionnement des constellations

Pour rendre des séquences de signaux plus résistantes au bruit, il est nécessaire qu'elles soient très différentes les unes des autres, autrement dit que la distance euclidienne entre les différents signaux pris dans leur espace soit la plus grande possible. C'est ici qu'interviennent réellement les MCT car elles permettent d'augmenter significativement la distance entre les séquences de signaux en comparaison avec une modulation non-codée.
Considérons une constellation initiale A_0 à 2^{m+1} points, de distance euclidienne minimale d_0, où chaque point est associé à un mot binaire de $(m+1)$ bits. La constellation A_0 peut être partitionnée en deux sous-constellations B_0 et B_1 possédant chacune 2^m points ainsi qu'une même distance euclidienne minimale $d_1 > d_0$.
Affectons à chaque point de la constellation B_0 (resp. B_1) un mot binaire dont le bit de poids faible est égal à 0 (resp. 1). L'opération de partitionnement peut être poursuivie et ainsi, à B_0 (resp. B_1) on peut associer deux sous-constellations C_0 et C_2 (resp. C_1 et C_3) à 2^{m-1} points possédant la même distance euclidienne minimale $d_2 > d_1$. Chaque point de la sous-constellation C_0 (resp. C_2) est codé par un mot binaire dont les deux bits de poids faible sont 00 (resp. 10). De la même façon, les points

des sous-constellations C_1 et C_3 sont respectivement codés par des mots dont les deux bits de poids faible sont 01 et 11.
En poursuivant le partitionnement, on obtient finalement 2^m sous-constellations contenant chacune deux points distants de $d_i > d_{i-1}$, chaque point étant codé par un mot binaire dont le bit de poids fort est différent.
Le concept global du partitionnement peut donc se traduire comme suit : "construire des sous-constellations dont le nombre de points est divisé par deux à chaque étape et dont la distance euclidienne minimale est croissante".
Ce principe de partitionnement est illustré sur les figures 1.9 et 1.10 : dans le premier cas avec la constellation circulaire d'une MDP-8 et dans le second cas avec la constellation carrée d'une MAQ-16.
Détaillons un peu plus l'exemple de la MDP-8 (cf. figure 1.9), celui-ci étant réutilisé par la suite.
La constellation circulaire A_0 possède $M = 8$ points et a pour distance minimale $d_0 = 2\sqrt{E}\sin(\pi/8)$ (où E est l'énergie utilisée pour transmettre un point de la constellation). Elle est partitionnée en deux sous-constellations B_0 et B_1 constituées chacune de 4 points, possédant la même distance minimale $d_1 = \sqrt{2E}$ qui correspond par ailleurs à la distance minimale de la constellation d'une modulation MDP-4 non-codée.
La sous-constellation B_0 (resp. B_1) engendre deux nouvelles sous-constellations C_0 et C_2 (resp. C_1 et C_3) possédant 2 points et de distance minimale $d_2 = 2\sqrt{E}$, ce qui correspond à la distance minimale de la constellation d'une modulation MDP-2 non-codée.

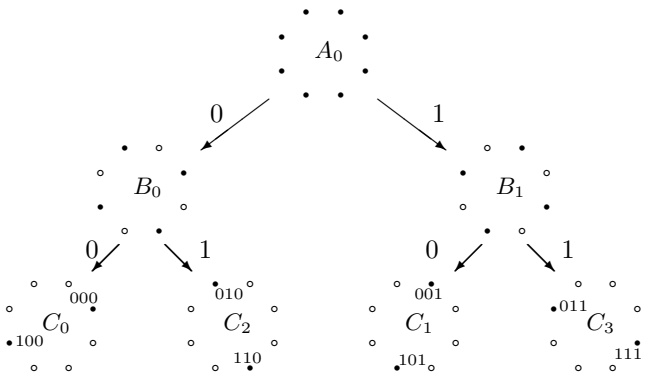


FIG. 1.9 – Partitionnement de la constellation MDP-8

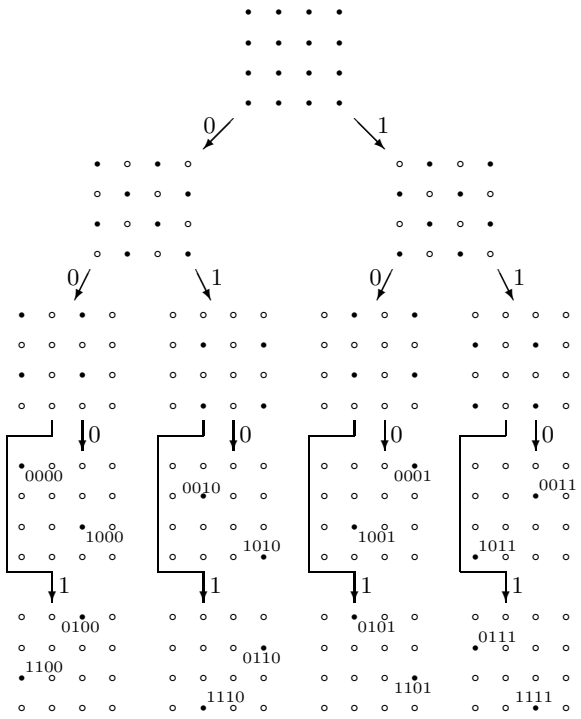


FIG. 1.10 – Partitionnement de la constellation MAQ-16

1.4.4 Principe de codage

Nous avons vu que pour les MCT, le modulateur émet des suites de signaux corrélés appartenant à une constellation possédant $M = 2^{m+1}$ points. Selon Ungerboeck [89], quelle que soit l'efficacité spectrale considérée pour la transmission et pour un code aussi complexe soit-il, le gain asymptotique de codage procuré par une MCT est quasi maximal en utilisant un seul élément binaire de redondance par symbole transmis(ou ce qui est équivalent par signal transmis). Ainsi pour une MCT construite à partir d'une constellation à $M = 2^{m+1}$ points, l'efficacité spectrale de la transmission est de $m\ bit/s/Hz$ et les performances de la MCT sont à comparer avec celles d'une modulation à 2^m états, c'est-à-dire possédant une constellation à 2^m points. La constellation d'une MCT a donc deux fois plus de points que celle de la modulation non-codée ayant la même efficacité spectrale.

Supposons donc que l'on veuille transmettre un bloc de m éléments binaires issus de la source d'information. On le scinde en deux blocs de longueur respectives n et $(m-n)$. Le bloc de longueur n est alors codé à l'aide d'un codeur convolutif de rendement $n/(n+1)$, le second bloc demeurant lui inchangé. Les $(n+1)$ éléments binaires issus du codeur sont ensuite utilisés pour sélectionner une sous-constellation à 2^{m-n} points tandis que les $(m-n)$ bits non codés permettent alors de choisir un point en particulier dans cette sous-constellation.

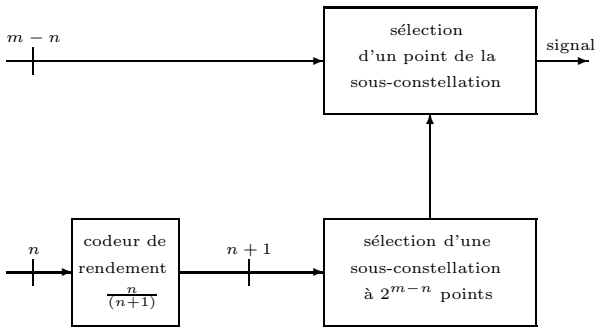


FIG. 1.11 – Schéma de principe d'une MCT

1.4.5 Construction du treillis d'une MCT

La réalisation du décodeur nécessite de construire au préalable un treillis de la MCT. Pour construire un tel treillis, quelques règles doivent être observées si l'on souhaite maximiser la distance libre. A cet effet, Ungerboeck propose les trois règles suivantes :

Règle 1 : Les $M = 2^{m+1}$ signaux de la constellation initiale (non patitionnée) doivent être utilisés avec la même fréquence.

Règle 2 : Les 2^{m-n} branches parallèles, si elles existent doivent être associées à des signaux appartenant à une même sous-constellation à 2^{m-n} points.

Règle 3 : Les 2^m branches qui quittent un état ou atteignent un état doivent être associées à des signaux appartenant à une même sous-constellation à 2^m points.

La règle 1 assure au treillis un motif régulier, les règles 2 et 3 garantissent pour leur part que la distance libre de la MCT est toujours supérieure à la distance euclidienne minimale de la modulation non-codée prise comme référence pour le calcul du gain de codage. On peut rappeler ici que dans

le cas d'une transmission de m symboles binaires, on compare une MCT à $2^{(m+1)}$ points dans la constellation initiale avec une modulation non-codée à 2^m points. Concrètement, en utilisant les règles 2 et 3, la distance euclidienne entre deux chemins qui divergent puis convergent sur une branche (cas des branches parallèles) ou sur plusieurs branches (au moins deux) est toujours supérieure à la distance minimale d'une sous-constellation à 2^m points. Il est à noter également que le nombre d'états du treillis est important comme le montre l'exemple qui suit.
Considérons la MCT associée au partitionnement de la figure 1.9. Si l'on note E l'énergie nécessaire pour transmettre un symbole (deux bits d'information et un bit de redondance), le rayon de la constellation initiale est alors $\sqrt{E}$. L'efficacité spectrale de cette MCT est donc de 2 $bit/s/Hz$ et ses performances sont à comparer à celles de la modulation MDP-4 (de distance euclidienne minimale $\sqrt{2E}$). Regardons ce qui se passe sur deux treillis distincts mais associés à une même MCT. Par convention, la notation (α, β) sur une branche reliant deux états signifie qu'il existe en fait deux branches entre ces deux états, l'une portant le label α et l'autre le label β.

- ◇ Treillis à deux états
  Dans ce cas on obtient le treillis de la figure 1.12 et l'on voit immédiatement que l'on ne parvient pas à mettre en pratique la règle 3. En effet les signaux 0 et 4 d'une part et les signaux d'autres part, convergent vers l'état 0, mais ils n'appartiennent pas à la même sous-constellation à 2^2 points.

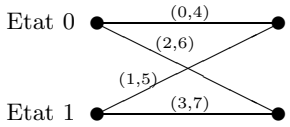


FIG. 1.12 – Treillis à 2 états d'une MCT à 8 états de phase

On vérifie que la distance libre de cette MCT est égale à la distance euclidienne entre les chemins représentés sur la figure 1.13 et qui correspondent à deux chemins qui divergent à un instant t et qui reconverge à l'instant $t + 2$.

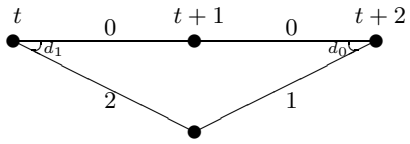


FIG. 1.13 – Chemins à la distance minimale pour un treillis à 2 états

La distance libre de la MCT à huit états de phase, pour un treillis à deux états, est donc :

$$d_{libre} = \sqrt{d_1^2 + d_0^2} = \sqrt{2E\left(1 + 2\sin^2\left(\frac{\pi}{8}\right)\right)}$$

En raisonnant à même énergie E utilisée pour transmettre deux éléments binaires d'information, le gain asymptotique de codage par rapport à une modulation MDP-4 est de :

$$G_{dB} = 10\log\left(\frac{d_{libre}^2}{(d_{min}^e)^2}\right) = 10\log\left(\frac{d_1^2 + d_0^2}{d_1^2}\right) = 10\log\left(1 + \frac{d_0^2}{d_1^2}\right) = 1.1\ dB$$

◇ Treillis à quatre états
Pour augmenter la distance libre de la MCT, il est nécessaire d'utiliser un treillis à quatre états qui permet alors de respecter les trois règles de bases proposées par Ungerboeck. Un tel treillis est proposé sur la figure 1.14. Dans ce cas, on peut vérifier que la distance libre est égale à la distance entre deux branches parallèles. Autrement dit :

$$d_{libre} = d_2 = 4E$$

Le gain asymptotique de codage par rapport à une modulation MDP-4 est donc de :

$$G_{db} = 10\log\left(\frac{d_{libre}^2}{(d_{min}^e)^2}\right) = 10\log\left(\frac{d_2^2}{d_1^2}\right) = 3\ dB$$

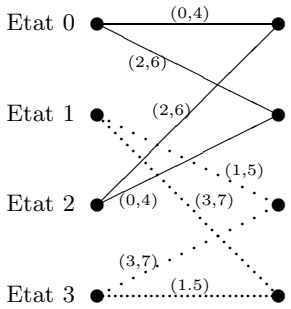


FIG. 1.14 – Treillis à 4 états d'une MCT à 8 états de phase

◇ On peut faire ici deux remarques. Tout d'abord, il est possible d'éviter les branches parallèles tout en respectant les trois règles de base proposées par Ungerboeck. Pour cela, il est nécessaire de construire un treillis à huit états qui permet alors d'obtenir une distance libre égale à :

$$d^2_{libre} = 4E\left(1 + \sin^2\left(\frac{\pi}{8}\right)\right)$$

Dans ce cas, le gain asymptotique G_{dB} par rapport à une modulation MDP-4 est de l'ordre de 3.6 dB. Ce gain de 0.6 dB par rapport au treillis précédent ne parvient pas à justifier le fait de doubler le nombre d'états du treillis de la MCT qui implique une complexité plus grande. De plus, sur les exemples ci-dessus, la distance libre de la MCT se détermine "à vue" sans réelle difficulté. En revanche, pour des treillis excédant huit états il faut avoir recours à des programmes que l'on peut par exemple trouver dans l'article de Biglieri et al. [14].

Les modulations codées sont un axe de recherche très intéressant notamment si l'on considère des codes correcteurs d'erreurs sur différents anneaux ou corps finis de façon à mettre en correspondance directe un élément d'un mot de code avec un élément de la constellation. La question posée est de savoir si un bon code code correcteur sur une extension de corps ou sur un anneau va nécessairement conduire à l'obtention d'une modulation codée ayant une bonne distance euclidienne. Une autre approche à envisager serait de concaténer des MCT entre elles *via* des entrelaceurs comme dans le cas des turbo codes, mais en prenant soin de ne pas "casser" les éléments pour préserver les critères de construction des MCT.

Chapitre 2

Décodage des codes correcteurs d'erreurs

Dans ce chapitre, nous abordons la notion de décodage des codes correcteurs d'erreurs. Les méthodes utilisées ici sont soit "souples" soit "dures" (Cf. Annexe) et sont appliquées à différentes constructions qu'elles soient graphiques (arbres, treillis,...) ou non (méthode de Chase,...). Dans un premier temps, la notion d'information extrinsèque est rappelée car elle est au cœur de tous les sytèmes de décodage actuels basés sur des traitements itératifs.

2.1 Notion d'information extrinsèque

Le processus de décodage itératif consiste à utiliser dans chacun des décodeurs mis en relation un algorithme de décodage qui détermine les APPs des symboles d'information :

$$APP = \text{Observation} \times APriori \times \text{information extrinsèque}$$

Plus généralement les algorithmes utilisés déterminent des valeurs de vraisemblance pour les symboles d'information. La séquence de valeurs de vraisemblance ainsi obtenue est alors transmise au décodeur suivant qui en fait de même. De cette façon, chaque décodeur tire profit des informations qui lui sont apportées par les autres décodeurs. Pour améliorer ces décisions chacun des décodeurs doit fonctionner avec de l'information qui n'a pas été émise par lui même comme le montrent Berrou et al. [12].
Dans le cas binaire, on utilise fréquemment le LLR ("Log-Likelihood Ratio")

défini par

$$L(u) = \ln \frac{Pr(u = +1|\text{entrée})}{Pr(u = -1|\text{entrée})}$$

Les LLR peuvent être calculés de façon exacte à l'aide de l'algorithme $BCJR$ [6] qui a cependant le défaut d'être plus complexe que l'algorithme de Viterbi [94] également présenté par Forney dans [40]. En contre partie pour le $SOVA$ [94], les estimations des vraisemblances sont en fait des approximations des LLRs.
Que ce soit l'algorithme $BCJR$ ou le $SOVA$, il existe dans la littérature essentiellement deux méthodes pour extraire l'information extrinsèque de chacun des décodeurs. Dans la première méthode, l'information extrinsèque à l'entrée d'un décodeur est modélisée comme étant la sortie d'un canal gaussien [12]. Dans la seconde méthode, l'information extrinsèque est utilisée pour mettre à jour les probabilités *a priori* qui seront utilisées pour la prochaine étape (ou itération) du décodage. De ce fait, les probabilités *a posteriori* calculées pour un décodeur deviennent les probabilités *a priori* pour un autre décodeur.

2.2 Décodage sur un arbre

Les différentes méthodes de décodage sur un arbre ont notamment été étudiées par Wiberg [99].
Dans un premier temps, fixons quelques notations destinées à simplifier l'expression des différentes règles de calcul :

- ◇ À un bit est associé le couple $b = (b(0), b(1))$ où $b(i)_{i=0,1}$ correspond à la vraisemblance que la valeur du bit soit égale à i.
- ◇ "*op*" représente le "sommet opérateur" qui permet de mettre en relation deux ou plusieurs symboles d'un même mot de code. Il correspond à l'opération effectuée entre deux bits reliés entre eux par l'opérateur ou-exclusif "$\oplus$" lors du codage.

2.2.1 Méthode min-somme

Soit $\mu_{b\to op}$ le message envoyé du bit b vers l'opérateur op.
La valeur de ce message se détermine comme suit :

$$\begin{bmatrix} \mu_{b\to op}(0) \\ \mu_{b\to op}(1) \end{bmatrix} = c + \begin{bmatrix} \sum_k \mu_{op_k\to b}(0) \\ \sum_k \mu_{op_k\to b}(1) \end{bmatrix}$$

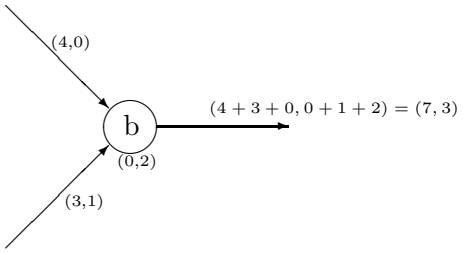
où c est une constante arbitraire et k un indice permettant d'énumérer les "sommets opérateurs", autre que celui vers lequel b propage l'information,

qui sont incidents à b.

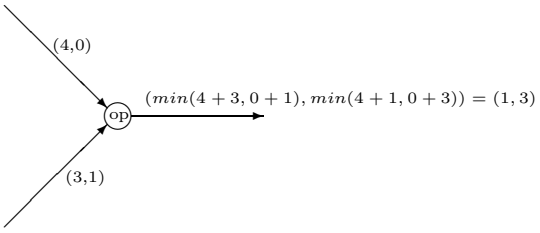
Soit maintenant $\mu_{op \to b}$ le message envoyé de l'opérateur op vers le bit b.
L'expression de la valeur de ce message est la suivante :

$$\begin{bmatrix} \mu_{op\to b}(0) \\ \mu_{op\to b}(1) \end{bmatrix} = c + \begin{bmatrix} min(\mu_{b_1\to op}(0) + \mu_{b_2\to op}(0), \mu_{b_1\to op}(1) + \mu_{b_2\to op}(1)) \\ min(\mu_{b_1\to op}(0) + \mu_{b_2\to op}(1), \mu_{b_1\to op}(1) + \mu_{b_2\to op}(0)) \end{bmatrix}$$

Regardons comment cela fonctionne sur un exemple.
Supposons qu'un bit b incident à deux opérateurs propage l'information reçue. Nous sommes dans la configuration suivante :



Supposons maintenant le second cas de figure où l'information en provenance de deux bits se propage à travers un opérateur :



2.2.2 Méthode somme-produit

Soit $\mu_{b\to op}$ le message envoyé du bit b vers l'opérateur op.
La valeur de ce message se détermine comme suit :

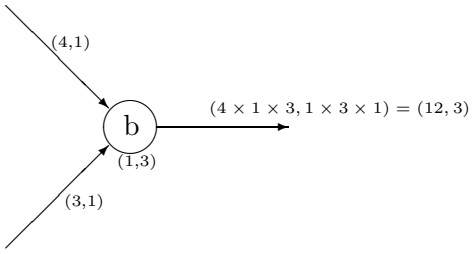
$$\begin{bmatrix} \mu_{b\to op}(0) \\ \mu_{b\to op}(1) \end{bmatrix} = \gamma \times \begin{bmatrix} \prod_k \mu_{op_k\to b}(0) \\ \prod_k \mu_{op_k\to b}(1) \end{bmatrix}$$

où γ est un facteur scalaire arbitraire non nul et k un indice permettant d'énumérer les "sommets opérateurs", autre que celui vers lequel b propage l'information, qui sont incidents à b.

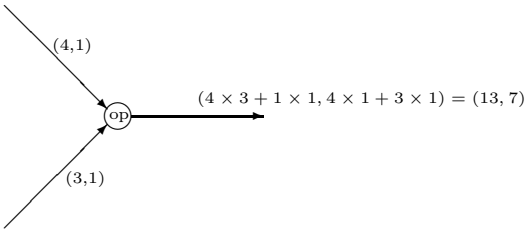
Soit maintenant $\mu_{op\to b}$ le message envoyé de l'opérateur op vers le bit b.
L'expression de la valeur de ce message est la suivante :

$$\begin{bmatrix} \mu_{op\to b}(0) \\ \mu_{op\to b}(1) \end{bmatrix} = c + \begin{bmatrix} (\mu_{b_1\to op}(0) \times \mu_{b_2\to op}(0)) + (\mu_{b_1\to op}(1) \times \mu_{b_2\to op}(1)) \\ (\mu_{b_1\to op}(0) \times \mu_{b_2\to op}(1)) + (\mu_{b_1\to op}(1) \times \mu_{b_2\to op}(0)) \end{bmatrix}$$

Regardons comment cela fonctionne sur un exemple.
Supposons qu'un bit b incident à deux opérateurs propage l'information reçue. Nous sommes dans la configuration suivante :



Supposons maintenant le second cas de figure où l'information en provenance de deux bits se propage à travers un opérateur :



2.2.3 Simplification des méthodes

Ces deux méthodes étant maintenant connues, il n'est pas inutile de savoir qu'elles se prêtent à certaines simplifications.
En particulier, pour l'exemple de la section suivante, c'est la méthode mS

qui est utilisée mais dans une version simplifiée.
En effet le vecteur message $\mu = (\mu(0), \mu(1))$ peut se "réduire" à une unique valeur réelle λ donnée par :

$$\lambda = \mu(1) - \mu(0)$$

Avec les notations définies précédemment, les règles de calcul deviennent alors :

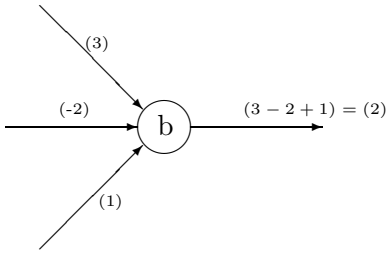
$$\lambda_{b \to op} = \sum_k \lambda_{op_k \to b}$$

$$\lambda_{op \to b} = \left(\prod_k signe(\lambda_{b \to op_k}) \right) \times \min_k (|\lambda_{b \to op_k}|)$$

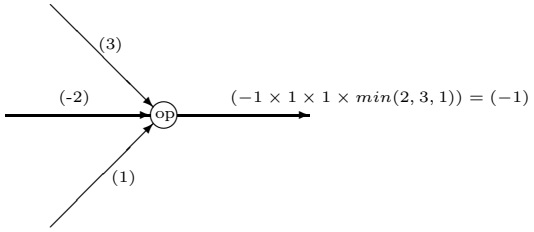
Attention : cette simplification des règles de calcul pour le min-somme ne peuvent s'effectuer que si les deux conditions suivantes sont vérifiées :

* Les variables sont binaires.
* Les opérateurs sont des opérateurs de parité.

Observons ces simplifications sur l'exemple suivant.
Supposons qu'un bit b incident à trois opérateurs propage l'information reçue. Nous sommes dans la configuration suivante :



Supposons maintenant le second cas de figure où l'information en provenance de trois bits se propage à travers un opérateur :



On remarque également que ces simplifications se révèlent d'autant plus intéréssantes que le nombre de branches incidentes à la "feuille" que l'on veut traiter sont importantes.
En effet ici on peut traiter simultanément toutes les branches incidentes au sommet, alors que dans le cas des règles initiales on est contraint de grouper les branches par deux.
Aussi si les hypothèses sont propices à la simplification des règles de calcul, on y a systématiquement recours, comme c'est le cas dans l'exemple à venir qui, malgré sa simplicité, a cependant le mérite de mettre en valeur quelques unes des méthodes entrevues dans cette section.

2.2.4 Exemples

2.2.4.1 Exemple 1 : cas défavorable

Dans le cadre de cette exemple, nous considérons le code de Hamming de paramètres [7,4,3] de matrice génératrice non systématique

$$G_{[7,4,3]} = \begin{pmatrix} 1&1&0&0&0&0&1 \\ 0&0&1&1&0&0&1 \\ 0&0&0&0&1&1&1 \\ 0&1&0&1&0&1&0 \end{pmatrix}$$

et de matrice de contrôle

$$H_{[7,4,3]} = \begin{pmatrix} 1&1&1&1&0&0&0 \\ 0&0&1&1&1&1&0 \\ 1&0&1&0&1&0&1 \end{pmatrix}$$

On en déduit les trois relations de parités suivantes :

$$\begin{cases} X_1 \oplus X_2 \oplus X_3 \oplus X_4 &= 0 \\ X_3 \oplus X_4 \oplus X_5 \oplus X_6 &= 0 \\ X_1 \oplus X_3 \oplus X_5 \oplus X_7 &= 0 \end{cases}$$

Ce code a la particularité de posséder une représentation sous forme d'un graphe de Tanner qui n'est pas trop complexe (figure 2.1) et qui est de ce fait facile à étudier.
L'autre aspect intéressant de ce code est que son graphe possède des cycles qui sont l'une des contraintes principales du décodage sur une structure graphique. Cette contrainte est d'ailleurs d'autant plus ennuyeuse que les codes considérés comme "bons" possèdent tous des cycles.
Si l'on tente de réaliser un décodage itératif sur la structure de la figure 3,

on est rapidement confronté à la question de savoir comment réinjecter les informations pour la seconde itération, sans bien sûr réinjecter l'information collectée sur un bit sur ce même bit car alors l'information serait faussée.
Une tentative de solution à ce problème, proposée par Farrell [36], consiste à "éclater" le graphe de façon à couper les cycles (figures 2.2 et 2.3), nous fournissant ainsi une structure graphique adaptée au décodage itératif.

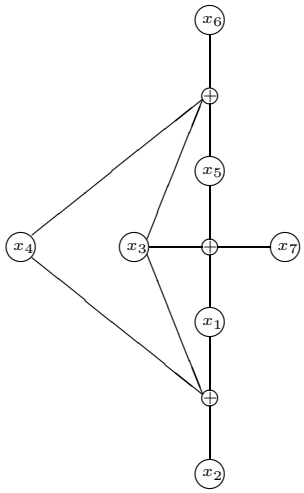


FIG. 2.1 – Graphe de Tanner du code de Hamming [7,4,3]

Considérons maintenant plus attentivement notre exemple.
Ici, les "$\oplus$" correspondent à une relation de parité et les "$\bigcirc$" à un bit.
Pour les régles de calcul destinées à gérer la propagation de l'information sur le graphe, nous faisons appel dans ce cas à la méthode min-somme simplifiée.

On suppose que la suite de symboles reçus est 0111010. En terme de métrique, à l'entrée du décodeur, le bit 0 est remplacé par une métrique de 4 et le 1 par une métrique de -4.
Lors de la première itération les métriques sont distribuées comme suit :

(1) 4 en X_1.
(2) -4 en X_2.
(3) Comme X_3 apparaît trois fois dans le graphe, chacun des ${X_3}^i$ se voit crédité de la métrique $\dfrac{-4}{3}$.
(4) Comme X_4 apparaît deux fois dans le graphe, chacun des ${X_4}^i$ se voit crédité de la métrique $\dfrac{-4}{2} = 2$.
(5) 4 en X_5.

(6) -4 en X_6.
(7) 4 en X_7.

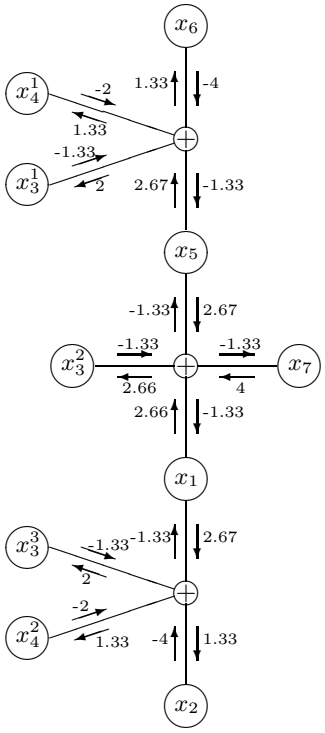


FIG. 2.2 – Décodage du Hamming [7,4,3] : itération 1

Au terme de la première itération, les métriques de X_1, X_2, $(X_3{}^i)_{i=1..3}$, $(X_4{}^i)_{i=1,2}$, X_5, X_6, X_7 ont pour valeurs respectives 1,33 ; -2,67 ; 0,67 ; 1,33 ; 0,67 ; -0,67 ; -0,67 ; 1,33 ; -2,67 ; 2,67.

Détaillons quelque peu ces valeurs en considérant par exemple le cas de X_7.
La nouvelle métrique de cet élément est 2,67. En fait, cette valeur est la somme de la métrique initiale de X_7, à savoir 4, avec la métrique "remontant" vers X_7, c'est-à-dire la métrique issue de l'interaction entre les différents X_i via le graphe de Tanner (-1,33 pour X_7).
D'où la nouvelle métrique de X_7 ($\mathcal{M}(X_7)$) : $\mathcal{M}(X_7) = 4 + (-1{,}33) = 2{,}67$.
Pour les $(X_3{}^i)_{i=1...3}$ et $(X_4{}^i)_{i=1,2}$ la démarche est similaire, exception faîte que, pour retrouver la valeur globale récoltée sur X_3, on somme les valeurs récoltées sur les $(X_3{}^i)_{i=1...3}$, d'où : $\mathcal{M}(X_3) = 0{,}67 + 1{,}33 + 0{,}67 = 2{,}67$.

De même, nous avons pour X_4 : $\mathcal{M}(X_4)$ = -0,67 + (-0,67) = -1,33.
Pour la seconde itération, on réinjecte les nouvelles valeurs affectées aux $(X_i)_{i=1...7}$ de la même manière que lors de l'initialisation des métriques de la première itération. Nous avons ainsi :

(1) 1,33 en X_1.
(2) -2,67 en X_2.
(3) Comme X_3 apparaît trois fois dans le graphe, chacun des ${X_3}^i$ se voit crédité de la métrique 0.89 $\left(= \dfrac{2.69}{3}\right)$.
(4) Comme X_4 apparaît deux fois dans le graphe, chacun des ${X_4}^i$ se voit crédité de la métrique -0.67 $\left(= \dfrac{-1.33}{2}\right)$.
(5) 1,33 en X_5.
(6) -2,67 en X_6.
(7) 2,67 en X_7.

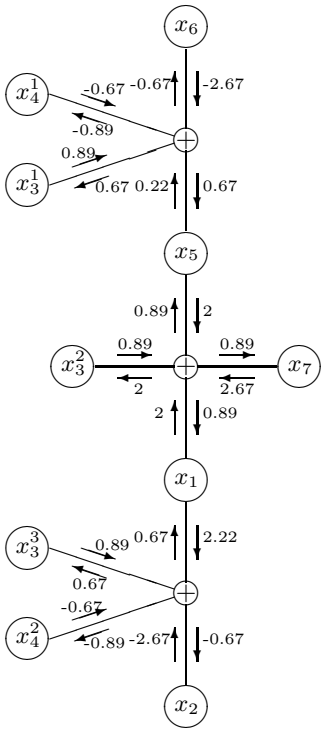


FIG. 2.3 – Décodage du Hamming [7,4,3] : itération 2

Au terme de la seconde itération, on suit une procédure identique à celle

utilisée à la fin de la premiére itération et l'on obtient ainsi les métriques finales des $(X_i)_{i=1\ldots7}$ qui sont respectivement 2,89 ; -3,34 ; 6,01 ; -3,12 ; 2,89 ; -3,34 ; 3,56.
On identifie alors les bits correspondants à ces métriques et l'on obtient ainsi le mot 0101010.
La suite de bits reçus étant 0111010, cela signifie que l'on a corrigé une erreur localisée sur le troisième bit.
On est ici en droit de se poser une question, à savoir : "pourquoi s'est-on arrêté après seulement deux itérations ?".
Une telle question ne rencontre pas pour écho, en général, de réponse simple comme le montrent Shao et al. [84] où sont mises en valeur diverses conditions d'arrêt de ces méthodes itératives.
En revanche sur l'exemple traité ci-dessus, la condition d'arrêt apparaît assez simplement.
Après la première itération, les métriques obtenues suffisent à corriger l'erreur sur le troisième bit car on constate que la métrique sur ce même bit est passée d'une valeur négative (représentative de la valeur binaire 1) à une valeur positive (représentative de la valeur binaire 0).
En fait, la seconde itération vient confirmer la convergence de la métrique sur le troisième bit qui passe de 2,67 à 6,01.
Les courbes de taux d'erreurs binaires sur un canal gaussien à bruit blanc additif pour une modulation MDP-2 sont données par la figure 2.4. La simulation est réalisée avec 5 itérations. Le problème majeur d'une telle représentation est que les différents bits n'ont pas la même fréquence d'apparition et la réinjection des extrinsèques devient très rapidement "aléatoire" car on ne sait comment la répartir en fonction du coefficient de duplication du bit dans l'arbre. Sur la courbe de décodage, ce phénomène se manifeste par un écart entre la courbe de décodage optimal et celle du décodage itératif qui ne cesse de croître lorsque le rapport signal à bruit augmente.

2.2.4.2 Exemple 2 : cas favorable

On considére le code de Hamming étendu [8,4,4] de matrice génératrice :

$$G_H = \begin{pmatrix} 1\,0\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0 \end{pmatrix}$$

Soit $m = (x_0, x_1, x_2, x_3, r_0, r_1, r_2, r_3)$ un mot de code où les $(x_i)_{i\in(0,\ldots,3)}$ (resp. $(r_i)_{i\in(0,\ldots,3)}$) correspondent aux bits d'information (resp. de redondance).

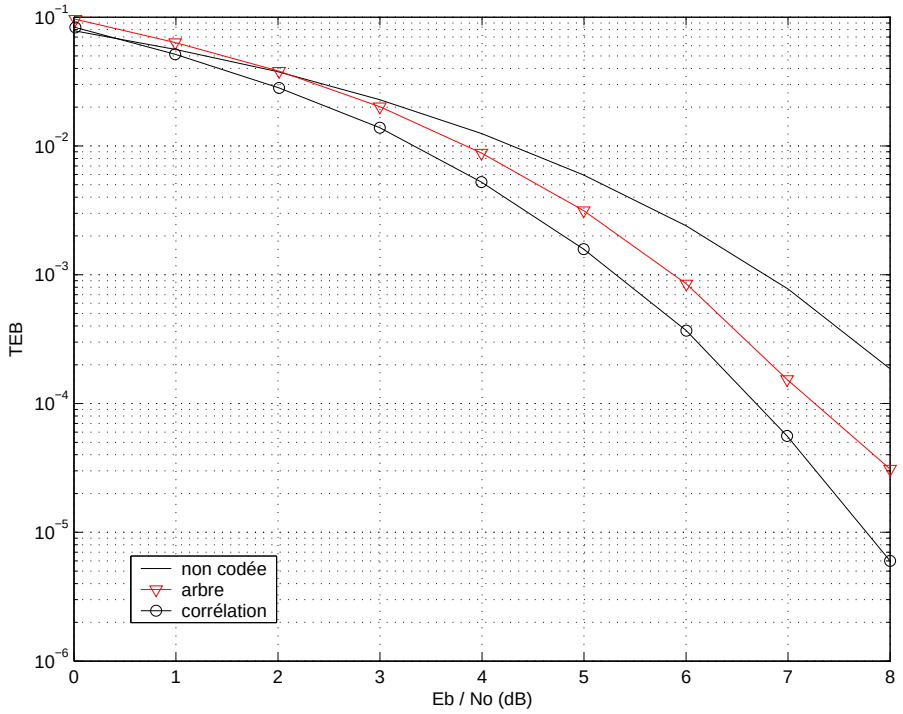


FIG. 2.4 – Décodage du Hamming [7,4,3]

Si $x_0 \oplus x_1 \oplus x_2 \oplus x_3 = 0$ alors $x = (x_0, x_1, x_2, x_3) = (r_0, r_1, r_2, r_3) = r$. Sinon, nous avons $\overline{x} = (\overline{x}_0, \overline{x}_1, \overline{x}_2, \overline{x}_3) = (r_0, r_1, r_2, r_3) = r$. On note a la variable intermediaire de complementation telle que $r = a \oplus x$. On peut alors construire le graphe de Tanner éclaté du code de Hamming (Cf. figure 2.5) dans lequel chaque bit a la même fréquence d'apparition (chaque bit intervient dans deux équations de parité).

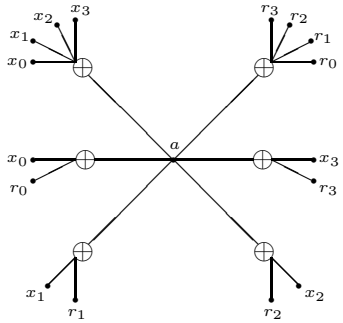


FIG. 2.5 – Graphe de Tanner éclaté du [8,4,4]

Soient α et β deux valeurs réelles. On définit la fonction de vraisemblance L, appelée aussi Log-Likelihood Ratio (LLR) par :

$$L(\alpha \oplus \beta) = |\alpha + \beta| - |\alpha - \beta|$$

et de façon récursive nous avons,

$$L(\alpha \oplus \beta \oplus \gamma \oplus \delta) = L(L(\alpha \oplus \beta) \oplus L(\gamma \oplus \delta))$$

Le décodage s'effectue alors en trois étapes.

- ◇ étape 1 :
  Calcul de la vraisemblance de la variable a telle que

$$a = \sum_{i=0}^{3} L(x_i \oplus r_i)$$

- ◇ étape 2 :
  Estimation des vraisemblances intermédiaires x'_i et r'_i des différents bits pour i variant de 0 à 3 :

$$\begin{aligned} x'_i &= x_i + signe(a) \times r_i \\ r'_i &= r_i + signe(a) \times x_i \end{aligned}$$

- ◇ étape 3 :
  Estimation des vraisemblances finales x''_i et r''_i :

$$\begin{array}{rclcrcl} x''_0 & = & x'_0 & + & L(a \oplus r'_0) & + & L(a \oplus x'_1 \oplus x'_2 \oplus x'_3) \\ r''_0 & = & r'_0 & + & L(a \oplus x'_0) & + & L(a \oplus r'_1 \oplus r'_2 \oplus r'_3) \\ & & & & & & \\ x''_1 & = & x'_1 & + & L(a \oplus r'_1) & + & L(a \oplus x'_0 \oplus x'_2 \oplus x'_3) \\ r''_1 & = & r'_1 & + & L(a \oplus x'_1) & + & L(a \oplus r'_0 \oplus r'_2 \oplus r'_3) \\ & & & & & & \\ x''_2 & = & x'_2 & + & L(a \oplus r'_2) & + & L(a \oplus x'_0 \oplus x'_1 \oplus x'_3) \\ r''_2 & = & r'_2 & + & L(a \oplus x'_2) & + & L(a \oplus r'_0 \oplus r'_1 \oplus r'_3) \\ & & & & & & \\ x''_3 & = & x'_3 & + & L(a \oplus r'_3) & + & L(a \oplus x'_0 \oplus x'_1 \oplus x'_2) \\ r''_3 & = & r'_3 & + & L(a \oplus x'_3) & + & L(a \oplus r'_0 \oplus r'_1 \oplus r'_2) \end{array}$$

Les courbes de taux d'erreurs binaires sur un canal gaussien à bruit blanc additif pour une modulation MDP-2 sont données par la figure 2.6. On remarque que contrairement à l'exemple précédent, la fréquence d'apparition des bits est ici la même pour chacun d'entre eux. Dans ce cas, la courbe représentant le décodage itératif reste très proche de celle du décodage optimal (environ 0.2 dB) et cette écart diminue lorsque l'on augmente le rapport signal à bruit.

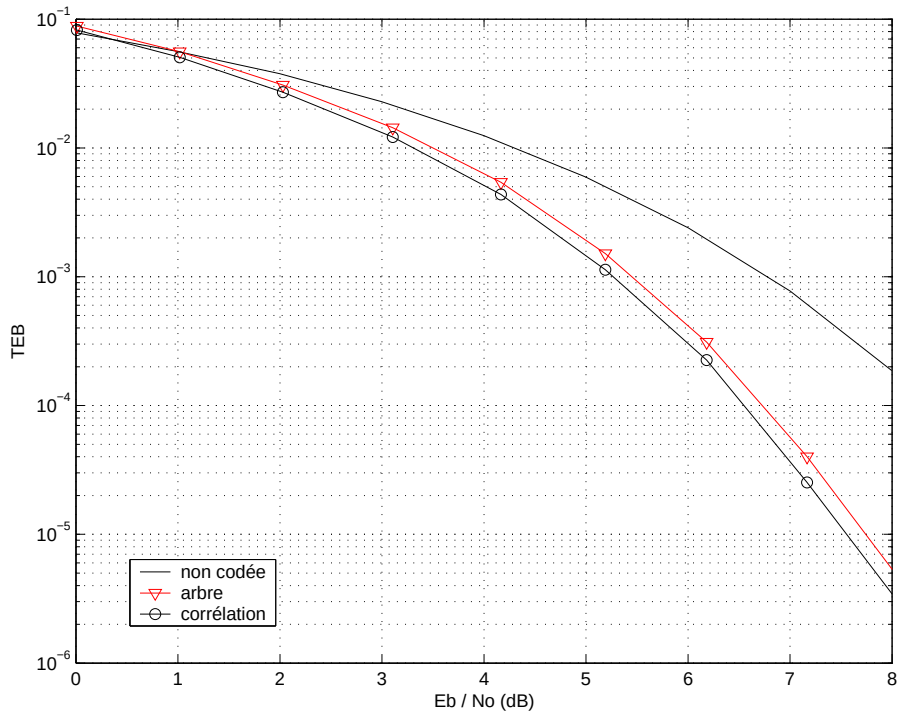


FIG. 2.6 – Décodage du Hamming [8,4,4]

2.3 Décodage sur treillis

Il ne s'agit pas dans cette section de faire un inventaire exhaustif de toutes les méthodes et nombreuses variantes qui ont été publiées mais de rappeler les algorithmes élémentaires qui ont servi de "support" à la réalisation des différents décodeurs implémentés durant cette thèse.

2.3.1 Méthode Forward-Backward

Cette méthode, appelée *BCJR* [6], permet d'estimer les probabilités *a posteriori* des états et des transitions d'une source de Markov observée à travers un canal discret, bruité et sans mémoire. On définit tout d'abord quelques notations.

- On note X_0^{N-1} une séquence $\{X_i\}$ émise par une source de Markov pour $0 \preceq i \prec N$.
- On note Y_0^{N-1} la séquence $\{Y_i\}$ reçue pour $0 \preceq i \prec N$.

- S_i est l'état de la source de Markov à l'instant i.
- $p_i(m'|m) = Pr(S_i = m'|S_{i-1} = m)$ est la probabilité de transition de l'état m à l'état m'.
- $q_i(X|m, m') = Pr(X_i = X|S_{i-1} = m; S_i = m')$ est la probabilité que la sortie soit X connaissant la transition.
- $p(Y_i|X_i)$ est la probabilité *a posteriori* de la sortie du canal Y_i connaissant l'entrée X_i.

Le but de cet algorithme est de traiter la séquence reçue Y_0^{N-1} pour estimer les $APPs$ des transitions :

$$\sigma(m, m') = Pr(S_{i-1} = m; S_i = m'|Y_0^{N-1}) = \frac{Pr(S_{i-1} = m; S_i = m', Y_0^{N-1})}{Pr(Y_0^{N-1})}.$$

On donne quelques notations supplémentaires :

- on note $\alpha_i(m) = Pr(S_i = m; Y_0^i)$ d'être en l'état m à l'instant i et d'avoir une observation "passée" Y_0^i de l'instant 0 à l'instant i.
- on note $\beta_i(m) = Pr(Y_{i+1}^{N-1}|S_i = m)$ la probabilité d'avoir une observation "future" Y_{i+1}^{N-1} de l'instant $i+1$ à l'instant $N-1$ sachant que l'état à l'instant i est l'état m.
- on note $\gamma_i(m, m') = Pr(S_i = m'; Y_i|S_{i-1} = m)$ la probabilité d'arriver à l'état m' et d'observer Y_i à l'instant i sachant que l'état à l'instant précédent était l'état m.

La source étant markovienne, si l'on considère que le treillis débute et s'achève en un même état 0, les $APPs$ se calculent comme suit :

Initialisation

$$\begin{aligned} \alpha_0(0) = 1 \quad \text{et} \quad \forall m \neq 0,\ \alpha_0(m) = 0 \\ \beta_{N-1}(0) = 1 \quad \text{et} \quad \forall m \neq 0,\ \beta_{N-1}(m) = 0 \end{aligned}$$

Procédure Forward

$$\forall i \in \{0, 1, \ldots, N-1\},\ \alpha_i(m') = \sum_m \gamma_i(m, m')\alpha_{i-1}(m)$$

Procédure Backward

$$\forall i \in \{N-2, \ldots, 1, 0\},\ \beta_i(m) = \sum_m{}' \gamma_{i+1}(m, m')\beta_{i+1}(m')$$

Calcul des *APPs*

$$\sigma_i(m,m') = \alpha_{i-1}(m) \times \gamma_i(m,m') \times \beta_i(m')$$

On note X_0^{N-1} la séquence de sortie d'un encodeur de rendement $\frac{k}{n}$, $X_i = (x_{i1}, \ldots, x_{in})$ la sortie à l'instant i et $Y_i = (y_{i1}, \ldots, y_{in})$ l'observation correspondante.
Les observations sur les différents bits étant indépendantes alors,

$$p(Y_i|X_i) = \prod_{j=1}^{n} p(y_{ij}|x_{ij})$$

Les $APP(x_{ic} = 0)$ proviennent de la somme des $\sigma_i(m,m')$ correspondant à l'ensemble des transitions pour lesquelles $x_{ic} = 0$. Par conséquent,

$$APP(x_{ic} = 0) = \sum_{m,m'|x_{ic}=0} \sigma_i(m,m') \tag{2.1}$$

Chaque transition ne comportant qu'une unique valeur de X_i, $q_i(X|m,m')$ ne possède qu'une seule valeur non nulle qui est égale à 1 (on considère le cas de l'anneau $\mathbb{Z}_2$). De plus, comme on ne dispose d'aucune information *a priori* sur les k bits d'information, la probabilité *a priori* $p_i(m'|m)$ est égale à 0.5^k. La probabilité *a posteriori* peut donc se mettre sous la forme :

$$APP(x_{ic} = 0) = 0.5^k \sum_{m,m'|x_{ic}=0} \alpha_{i-1}(m) \times \prod_{j=1}^{n} p(y_{ij}|x_{ij}) \times \beta_i(m') \tag{2.2}$$

2.3.2 Méthode Max-Log-Map

Cette méthode est en fait une approximation de la précédente. Pour cela on se place dans le domaine logarithmique en introduisant la notion de rapport de vraisemblance logarithmique (ou "Log-Likelihood Ration") défini par

$$LLR(X) = \ln\left(\frac{Pr(X=1)}{Pr(X=0)}\right)$$

où X est une variable aléatoire pouvant prendre la valeur 0 ou 1.
On note toujours X_0^{N-1} la séquence de sortie d'un encodeur de rendement $\frac{k}{n}$, X_i la sortie à l'instant i et Y_i l'observation correspondante. D'après l'équation (2.1), la LLR du bit codé x_{ic} s'écrit alors :

$$LLR(x_{ic}) = \ln\left(\frac{\displaystyle\sum_{m,m'|x_{ic}=1} \alpha_{i-1}(m) \times \gamma_i(m,m') \times \beta_i(m')}{\displaystyle\sum_{m,m'|x_{ic}=0} \alpha_{i-1}(m) \times \gamma_i(m,m') \times \beta_i(m')}\right) \tag{2.3}$$

Les procédures Forward et Backward sous forme normalisée s'écrivent alors :

Procédure Forward

$$\forall i \in \{0,1,\ldots,N-1\},\ \alpha_i(m') = \frac{\sum_m \gamma_i(m,m')\alpha_{i-1}(m)}{\sum_m \sum_{m'} \gamma_i(m,m')\alpha_{i-1}(m)} \quad (2.4)$$

Procédure Backward

$$\forall i \in \{N-2,\ldots,1,0\},\ \beta_i(m) = \frac{\sum_{m'} \gamma_{i+1}(m,m')\beta_{i+1}(m')}{\sum_m \sum_{m'} \gamma_{i+1}(m,m')\beta_{i+1}(m')} \quad (2.5)$$

Par passage dans le domaine logarithmique, les équations (2.4) et (2.5) deviennent :

$$\overline{\alpha}_i(m') = \ln(\alpha_i(m')) \text{ et } \overline{\beta}_i(m) = \ln(\beta_i(m))$$

d'où,

$$\overline{\alpha}_i(m') = \ln\left(\sum_m \gamma_i(m,m')\alpha_{i-1}(m)\right) - \ln\left(\sum_{m'}\sum_m \gamma_i(m,m')\alpha_{i-1}(m)\right) \quad (2.6)$$

$$\overline{\beta}_i(m) = \ln\left(\sum_{m'} \gamma_{i+1}(m,m')\beta_{i+1}(m')\right) - \ln\left(\sum_m \sum_{'m} \gamma_{i+1}(m,m')\beta_{i+1}(m')\right) \quad (2.7)$$

On introduit alors l'approximation suivante qui est à la base de la simplification de la méthode.

$$\ln(e^{\delta_1} + \ldots + e^{\delta_n}) \approx \max_{i\in\{0,\ldots,n\}} \delta_i \quad (2.8)$$

En appliquant cette approximation aux équations (2.6) et (2.7), on obtient :

$$\begin{aligned}\overline{\alpha}_i(m') &= \ln\left(\sum_m \overline{\gamma}_i(m,m')\overline{\alpha}_{i-1}(m)\right) - \ln\left(\sum_{m'}\sum_m \overline{\gamma}_i(m,m')\overline{\alpha}_{i-1}(m)\right)\\ &\approx \max_m\left(\overline{\gamma}_i(m,m') + \overline{\alpha}_{i-1}(m)\right) - \max_{m',m}\left(\overline{\gamma}_i(m,m') + \overline{\alpha}_{i-1}(m)\right)\end{aligned}$$

$$\begin{aligned}\overline{\beta}_i(m) &= \ln\left(\sum_{m'} \overline{\gamma}_{i+1}(m,m')\overline{\beta}_{i+1}(m')\right) - \ln\left(\sum_m\sum_{m'} \overline{\gamma}_{i+1}(m,m')\overline{\beta}_{i+1}(m')\right)\\ &\approx \max_{m'}\left(\overline{\gamma}_{i+1}(m,m') + \overline{\beta}_{i+1}(m')\right) - \max_{m,m'}\left(\overline{\gamma}_{i+1}(m,m') + \overline{\beta}_{i+1}(m')\right)\end{aligned}$$

où $\overline{\gamma}_i(m,m') = \ln(\gamma_i(m,m'))$.
De façon similaire, on peut approximer l'équation (2.3) par

$$\begin{array}{rcl} LLR(x_{ic}) & \approx & \max_{m,m'|x_{ic}=0} \left(\overline{\alpha}_{i-1}(m) + \overline{\gamma}_i(m,m') + \overline{\beta}_i(m')\right) \\ & & - \max_{m,m'|x_{ic}=1} \left(\overline{\alpha}_{i-1}(m) + \overline{\gamma}_i(m,m') + \overline{\beta}_i(m')\right) \end{array}$$

2.4 Décodage analogique

Actuellement, les techniques classiques de décodage font appel à des décodeurs numériques à temps discret implémentés sur des processeurs numériques ou des ordinateurs. Face au succès croissant des transmissions analogiques, la question de la nécessité de la transformation du signal analogique en élément numérique compatible avec le décodeur commence à apparaître. Hagenauer est l'un des premiers à apporter une réponse, notamment dans [54], où il propose l'utilisation directe du signal analogique reçu dans un décodeur lui aussi analogique mais à temps continu.
La présentation de ces décodeurs analogiques s'articule autour de deux points. Le premier consiste en quelques rappels, notamment sur les graphes de Tanner. Le second point s'attache quant à lui à une approche théorique de ces décodeurs. Finalement dans la dernière partie, un exemple de modélisation de décodeur analogique pour un code de Hamming de paramètres [7,4,3] est donné.

2.4.1 Rappels

En préambule à toute opération de décodage, il est nécessaire de se munir d'une structure graphique la mieux adaptée possible à cette procédure. Dans le cadre du décodage analogique, ce sont les graphes de Tanner que l'on utilise. Ils correspondent en fait à une représentation graphique des liens de parité entre les bits définis par la matrice H de contrôle de parité.
Considérons par exemple un code équivalent au code de Hamming [7,4,3] et dont une matrice de parité s'écrit sous la forme :

$$H_{[7,4,3]} = \begin{pmatrix} 1&1&0&1&1&0&0 \\ 1&0&1&1&0&1&0 \\ 0&1&1&1&0&0&1 \end{pmatrix}$$

Cette matrice définit trois relations de parité qui sont les suivantes :

$$\begin{cases} X_0 \oplus X_1 \oplus X_3 \oplus X_4 = 0 \\ X_0 \oplus X_2 \oplus X_3 \oplus X_5 = 0 \\ X_1 \oplus X_2 \oplus X_3 \oplus X_6 = 0 \end{cases}$$

A l'aide de ces trois relations de parité nous pouvons alors dresser le graphe de Tanner représenté par la figure 2.7.

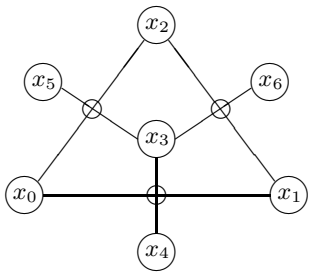


FIG. 2.7 – Graphe de Tanner du Hamming [7,4,3]

Une caractéristique apparente de ce graphe est qu'il possède plusieurs cycles($[x_4, x_3, x_6, x_1]$, $[x_5, x_3, x_4, x_0]$ et $[x_6, x_2, x_5, x_3]$), ce qui a priori n'en fait pas la structure la mieux adaptée au décodage. Cependant, dans la partie suivante, nous verrons que cela n'est plus un problème majeur pour un décodeur analogique qui parvient à fonctionner sur une structure comportant des cycles sans pour autant avoir de pertes significatives en terme de qualité de décodage.

2.4.2 Généralités

Dans le cadre de l'analogique, le décodage *SISO* ("Soft-In Soft-out") est réalisé par un réseau analogique non linéaire où les valeurs des bits sont représentées par des courants ou des tensions. Ici, les valeurs disponibles en sortie de canal servent à initialiser le réseau de décodage qui évolue alors de lui-même, continuement par rapport au temps, jusqu'à un état final correspondant à un état d'équilibre régi par les lois physiques des circuits électriques. Ainsi, les valeurs telles que tensions, différences de potentiel ou courants relevés en certains points du réseau constituent le décodage "soft" recherché.
Le premier aspect marquant d'un tel système est sa non-linéarité due à la présence de transistors que l'on modélisera par la suite au moyen de fonctions non-linéaires telles que des exponentielles ou des tangentes hyperboliques.
On touche d'ailleurs ici à l'un des points sensibles du décodage analogique car, par définition même, il s'effectue sur un réseau électronique et est plus difficile à simuler à l'aide d'outils numériques. Pour ces réalisations, il serait

plus commode de disposer d'un simulateur électrique analogique de type SPICE par exemple. Pour remédier à ce problème, Hagenauer propose, notamment dans [53], de simuler ces réseaux à l'aide d'applications purement numériques.
En fait, on peut distinguer ici deux types de décodages distincts, l'un algébrique et l'autre probabiliste selon la nature des applications utilisées pour la simulation.
Le décodage algébrique, essentiellement utilisé pour fournir une forte protection contre les petits niveaux de bruit, repose sur des techniques d'algèbre abstraite alors que le décodage probabiliste, plutôt approprié pour fournir une protection modérée contre les hauts niveaux de bruit, repose sur des techniques de décision statistique.
Plus concrètement, considérons le cas où l'on désire encoder deux éléments u_1 et u_2.
A cet effet, on utilise la structure de la figure 2.8.
Après l'opération de codage, on récupère trois bits, x_1 et x_2, obtenus comme de simples copies des bits d'entrée, alors que x_3 provient du "ou exclusif" entre x_1 et x_2.

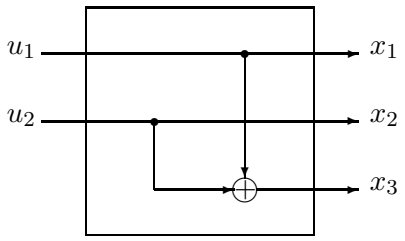


FIG. 2.8 – Codeur de parité

Pour décoder les trois bits ainsi générés, on peut procéder de deux façons différentes. On peut utiliser par exemple un décodeur algébrique (Cf. figure 2.9).

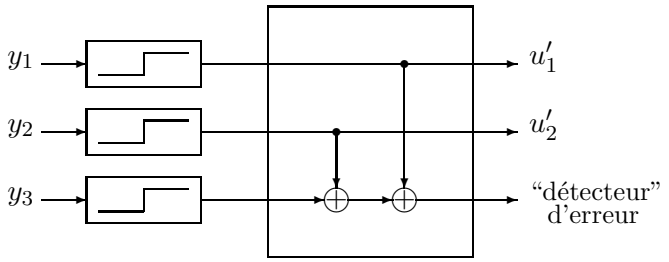


FIG. 2.9 – Décodeur "hard" du code de parité

Ici, à l'entrée du décodeur, les valeurs "soft", disponibles en sortie du canal de transmission, $(y_i)_{i=1...3}$ sont immédiatement transformées en valeurs "hard" qui vont être utilisées pour le décodage à proprement dit.
Une seconde manière d'opérer le décodage est d'utiliser un décodeur probabiliste (Cf. figure 2.10) qui lui travaille directement avec des valeurs "soft".

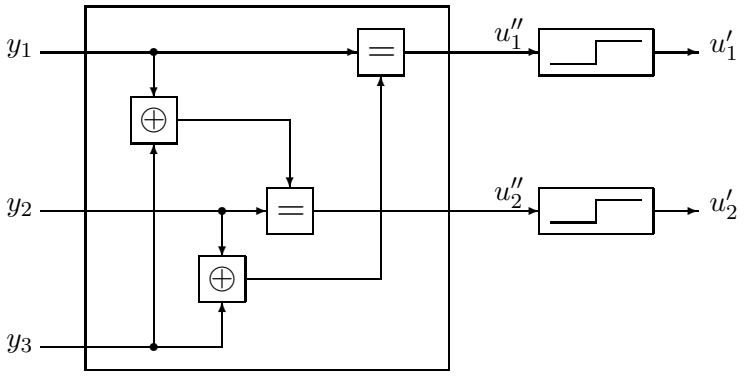
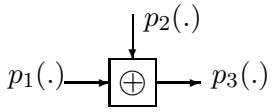


FIG. 2.10 – Décodeur "soft" du code de parité

Le décodeur algébrique est parfaitement adapté à une implémentation digitale car il implique essentiellement des opérations "ou exclusif". Cependant, la qualité du décodage est moindre en comparaison d'un décodeur probabiliste qui, lui, est plus complexe à implémenter, mais qui à l'avantage de fonctionner avec les valeurs analogiques reçues.

Détaillons maintenant le fonctionnement local de la méthode probabiliste, à savoir comment opèrent les boîtiers intervenant dans ce décodeur.
Mis à part les boîtiers de conversion des valeurs "soft" en valeurs "hard" intervenant à la fin du décodage, on rencontre deux autres types de boîtiers. Le premier se définit comme suit :

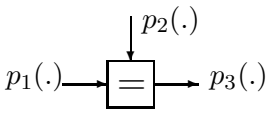


où $p_3(.)$ est obtenue à l'aide des deux relations suivantes :

$$p_3(0) = p_1(0) \times p_2(0) + p_1(1) \times p_2(1)$$

$$p_3(1) = p_1(0) \times p_2(1) + p_1(1) \times p_2(0)$$

Le second type de boîtier se définit quant à lui de la manière suivante :



où $p_3(.)$ est obtenue à l'aide des deux relations suivantes :

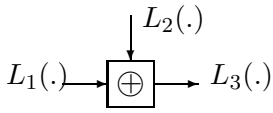
$$p_3(0) = \gamma \times p_1(0) \times p_2(0)$$

$$p_3(1) = \gamma \times p_1(1) \times p_2(1)$$

où γ est choisi tel que l'opération de normalisation "$p_3(0) + p_3(1) = 1$" soit toujours vérifiée.
Sur un réseau de décodage de codes tels que les turbo-codes ou les $LDPC$, de nombreux cycles apparaissent. Aussi, lors de l'implémentation digitale, si l'on opte pour un décodage itératif, on est amené à itérer plusieurs fois des calculs similaires en divers points du réseau jusqu'à atteindre un état stable. L'avantage (non démontré) du traitement analogique en temps continu sur le réseau est qu'il est censé éliminer ces itérations auxquelles se substitue le comportement général du réseau.
Ici on peut voir apparaître l'une des limites de la représentation numérique du décodage analogique, car si les deux boîtiers présentés précédemment permettent le décodage de codes $LDPC$, ils se révèlent bien souvent insuffisants pour le décodage des turbo-codes.
Une alternative à ce problème est la modifications de ces boîtiers via l'utilisation de deux nouveaux outils de calcul.
Le premier consiste en un nouveau type d'information qui se définit comme suit :

$$L(x) = log\left(\frac{p(x=0)}{p(x=1)}\right)$$

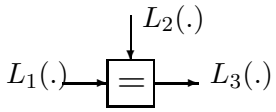
On parle dans ce cas de logarithme de rapport de vraisemblance encore appelé vraisemblance.
A l'aide de cette nouvelle définition, l'écriture de nos boîtiers peut se reformuler de la façon suivante :



où $L_3(.)$ est obtenue par la relation :

$$L_3 = 2 \times tanh^{-1}\left(tanh\left(\frac{L_1}{2} \right) \times tanh\left(\frac{L_2}{2} \right) \right)$$

Par la suite, on fera référence à ce boîtier sous l'appellation "*boxplus*" et l'on écrira $L(x_3) = boxplus(L(x_1), L(x_2))$.
Le second boîtier, que l'on nommera "*somme*", devient quant à lui :



où $L_3(.)$ est donnée par la relation "$L_3 = L_1 + L_2$".
En particulier, l'approche de Hagenauer du décodage analogique est en partie basée sur cette dernière représentation et sur la réalisation de ces deux boîtiers en tant que circuits analogiques. Par ailleurs, dans le cadre de l'exemple qui suit, ce sont ces outils de calcul qui sont utilisés pour les simulations.

2.4.3 Exemple

Nous considérons ici le code de Hamming [7,4] présenté dans la première partie et plus particulièrement son graphe de Tanner.
On conserve l'architecture générale de ce graphe mais on y intégre les boîtiers de la partie précedente, en remplaçant les opérateurs de parité par des "*boxplus*" et en intégrant les "*somme*" au niveau des feuilles symbolisant les bits. On obtient alors le graphe de la figure 2.11 :

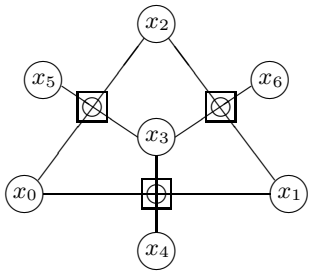


FIG. 2.11 – Graphe de Tanner modifié du Hamming [7,4,3]

Le graphe de Tanner ainsi modifié est alors utilisé comme structure graphique de décodage. Les éléments $(x_i)_{i=0...6}$ sont initialisés à l'aide

de la fonction $L(.)$ définie précédemment que l'on détermine pour chaque bit à partir des données reçues en sortie du démodulateur. Afin de faciliter la convergence du système, on ajoute à $L(x_3)$ la valeur $boxplus(boxplus(L(x_4), L(x_5)), L(x_6))$ qui en terme physique se traduit par le "quicklook-in property" du code. De façon plus algébrique, on peut remarquer que ce processus d'initialisation peut se déduire des trois relations de parité du code. En effet, on s'aperçoit en les combinant que seul le bit x_3 peut s'écrire comme combinaison binaire de bits de redondance, avec en l'occurence : $x_3 = x_4 \oplus x_5 \oplus x_6$.
Les différentes opérations sont réalisées simultanément en chaque endroit du graphe et chacun des bits $(x_i)_{i=0...6}$ propage l'intégralité de l'information qu'il possède dans les branches qui lui sont incidentes. La collecte de l'information au niveau de ces bits se traduit par une simple somme algébrique portant sur l'information déja détenue par le bit et celle lui arrivant.
De façon plus formelle, si I^t représente l'information portée par un bit à l'instant t et I_{in}^t l'information recueillie par un bit à l'instant t, nous avons la relation :

$$I^{t+1} = I^t + 0.6 \times I_{in}^{t+1}$$

La mise en application de la méthode correspond aux résultats de la figure 2.12 où le décodage est effectué sur un canal gaussien à bruit blanc additif utilisant une modulation MDP-2. On constate que la courbe du décodeur analogique est très proche de la courbe de décodage optimal et qu'elles deviennent de plus en plus proche lorsque le rapport signal à bruit augmente. Ces résultats sont tout de même à pondérer car ces courbes ont été obtenues à l'aide de programmes censés simuler l'analogique. Ces programmes sont donc numériques et fonctionnent par itérations, ce qui ne permet pas d'évaluer les capacités d'un décodeur analogique réellement implémenté dans une puce et qui ne serait tributaire que des courants ou des tensions.

2.5 Décodage sans support graphique

2.5.1 Décodage par syndrome

Cette méthode repose sur l'exploitation des propriétés d'orthogonalité d'un code avec son code dual. On peut ainsi obtenir des informations sur l'erreur *via* la projection du mot reçu sur l'espace dual (on appelle cette image un syndrome).
On considère un code $\mathcal{C}$ (de longueur n et de dimension k) de matrice génératrice G et de matrice de parité (ou de contrôle) H. On note $\mathcal{C}^\perp$ son

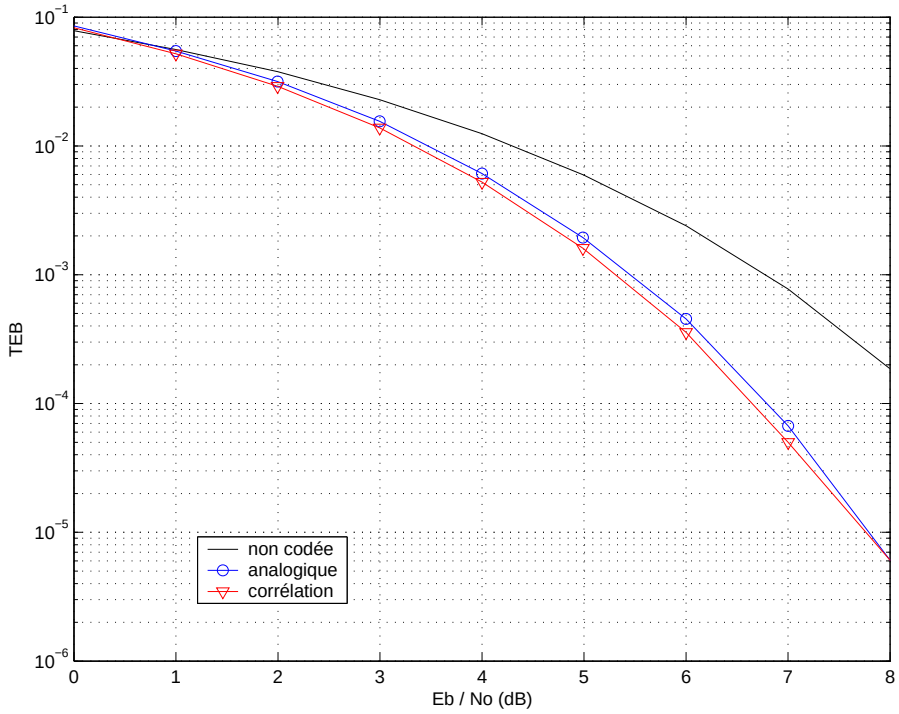


FIG. 2.12 – Décodage analogique du Hamming [7,4,3]

code dual de matrice génératrice H et de matrice de parité G.
On définit la relation d'équivalence

$$x\mathcal{R}y \Longleftrightarrow (x-y) \in \mathcal{C}$$

Les cosets d'un code $\mathcal{C}$ sont alors les classes d'équivalence de la forme $x+\mathcal{C}$ où x est un élément de $\mathbb{F}_2^n$.

Définition 2.5.1 *Soit H une matrice de contrôle pour $\mathcal{C}$. On définit l'application $\mathcal{S}$ par*

$$\left\{ \begin{array}{rccc} \mathcal{S}: & \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^{n-k} \\ & x & \longmapsto & H.{}^t x \end{array} \right.$$

$\mathcal{S}(x)$ est appelé le syndrome de x relativement à H.

Proposition 2.5.1

$$x \in \mathcal{C} \Longleftrightarrow H.{}^t x = 0$$

Remarque 2.5.1 *Deux éléments sont dans une même classe d'équivalence si et seulement si ils ont le même syndrome.*

Proposition 2.5.2 *Dans un coset donné, il existe au plus un élément x tel que $w(x) < \frac{d}{2}$.*

Définition 2.5.2 *Un élément de plus petit poids dans un coset est appelé "coset leader".*

Principe du décodage par syndrome :
On considère ici que le mot émis est x ($x \in \mathcal{C}$), l'erreur e et le message reçu y ($y = e + \mathcal{C}$ et y et e sont donc dans la même classe d'équivalence).

1. On crée le tableau standard du code qui associe à chaque syndrome tous les éléments du coset ainsi que le coset leader.
2. On calcule le syndrome de y qui est associé à la classe $y + \mathcal{C}$ et on choisit le coset leader y' de ce coset.
3. On décode au maximum de vraisemblance en considérant que le mot effectivement émis est $y + y'$.

Cette méthode, simple à mettre en œuvre, possède cependant l'inconvénient de ne plus être exploitable dès que la dimension du code dual devient trop importante (le nombre de coset est 2^{n-k}), rendant ainsi l'élaboration de la table des syndrome beaucoup trop coûteuse en terme de temps de calcul machine.

2.5.2 Décodage de Chase

Cette méthode proposée initialement par Chase dans [27] peut être appliquée à tout code en bloc possédant un décodeur algébrique. Si l'on considère un canal gaussien, cela revient à minimiser la distance euclidienne entre l'observation et un mot de code. Soient r l'observation réelle, y l'observation seuillée ($y \in \mathbb{F}_2^n$) et c un mot de code quelconque d'un code binaire

$\mathcal{C}[n,k]$. La distance euclidienne entre r et c est alors donnée par

$$\begin{aligned} d_E^2(r,c) &= \sum_{i=0}^{n-1}(r_i - c_i)^2 \\ &= ||r||^2 + n - 2\sum_{i=0}^{n-1}|r_i|y_i.c_i \\ &= ||r||^2 + n - 2\sum_{i=0}^{n-1}|r_i| + 4\sum_{\substack{i=0 \\ y_i \neq c_i}}^{n-1}|r_i| \end{aligned}$$

En mappant l'ensemble $\{-1,1\}$ sur l'ensemble $\{0,1\}$, l'égalité ci-dessus devient :

$$d_E^2(r,c) = ||r||^2 + n - 2\sum_{i=0}^{n-1}|r_i| + 4\sum_{i=0}^{n-1}(|r_i|y_i \oplus c_i)$$

Minimiser cette distance euclidienne revient donc à minismiser le terme $\sum_{i=0}^{n-1}(|r_i|y_i \oplus c_i)$ par rapport à l'ensemble des mots du code.

Le problème majeur d'une telle méthode étant le nombre élevé de mots de code, Chase a proposé de restreindre la recherche du mot de code le plus vraisemblable à un simple sous-ensemble de mots de code.

Dans un premier temps, l'observation réelle r est seuillée pour fournir l'observation binaire y. On détermine le mot de code $c^{(0)}$ le plus proche de y à l'aide d'un décodeur algébrique corrigeant au plus $t = \lfloor \frac{d-1}{2} \rfloor$ erreurs. $c^{(0)}$ est l'unique mot de code situé ł'intérieur de la boule de centre y et de rayon t. La méthode de Chase consiste à étendre la recherche à l'ensemble des mots de code appartenant à la sphère de centre y et de rayon $2t = d - 1$, le mot de code émis s'y trouvant forcément si le nombre d'erreurs commises ne dépasse pas la capacité correctrice du code. Cela revient à augmenter le pouvoir de correction du décodeur. On note $t^{(i)}$ les séquences de longueur n et de poids $\lfloor \frac{d}{2} \rfloor$ utilisées pour "atteindre" les mots de codes contenus dans la sphère de centre y et de rayon $\lfloor \frac{d}{2} \rfloor$. Pour chacun des $t^{(i)}$ on calcule la séquence $y^{(i)} = t^{(i)} \oplus y$ à laquelle on associe un mot de code $c^{(i)}$ qui est déterminé à l'aide du décodeur algébrique. Parmi tous les mots de codes ainsi construits, on choisit finalement celui qui vérifie $\min_j \sum_{i=0}^{n-1}(|r_i|y_i \oplus c_i^{(j)})$.

Chase a décliné son algorithme en 3 versions distinctes permettant ainsi

de diminuer le nombre de séquences de test nécessaires, réduisant donc la complexité, mais dégradant les performances du décodage.

- ◇ **Algorithme 1 :** Il s'agit de la version originale de l'algorithme explicitée ci-dessus. Il implique de former toutes les séquences test de poids $\lfloor \frac{d}{2} \rfloor$ pour obtenir tous les mots de code contenus dans la sphère $\mathcal{S}(y, d-1)$. Le coût de calcul est malheureusement rédhibitoire lorsque la distance minimale (ou la longeur) est importante car il nécessite la formation de $\mathrm{C}_n^{\lfloor \frac{d}{2} \rfloor}$ séquences de test.
- ◇ **Algorithme 2 :** On considère les $\lfloor \frac{d}{2} \rfloor$ indices correspondant aux valeurs les moins vraisemblables des $|r_i|$. On ne forme que les motifs d'erreur dont les 1 sont positionnés en ces indices. Le nombre de séquences de test est alors $2^{\lfloor \frac{d}{2} \rfloor}$.
- ◇ **Algorithme 3 :** Dans ce cas, le nombre de séquences de test est $\lfloor \frac{d}{2} + 1 \rfloor$ où chacune d'entre elles possède exactement i 1. Ces i positions correspondent aux i valeurs $|r_j|$ de vraisemblances les plus faibles pour $i \in \{0, 1, 3, \ldots, d-1\}$ si d est paire et $i \in \{0, 2, 4, \ldots, d-1\}$ dans le cas contraire.

Remarque 2.5.2 *Les deux dernières versions de l'algorithme sont très sensibles au nombre de positions considérées comme les moins fiables. On peut augmenter ce choix afin d'avoir un sous-ensemble de mots de code plus important et améliorer ainsi la probabilité que le mot de code effectivement émis lui appartiennent.*

Chapitre 3

Codes cortex

Cette famille de turbo codes en blocs est issue des travaux conjoints de Carlach et Vervoux [26]. De nombreux espoirs ont été placés dans cette construction et seront alimentés notamment par les travaux de Olocco et Tillich [73][74] et Otmani [77]. Leurs études concernaient plus particulièrement les aspects théoriques de la construction avec ses différentes propriétés, alors que l'axe principal adopté durant cette thèse est la difficulté de décoder cette famille de codes et la nécessité de modifier cette structure pour la rendre mieux adaptée au décodage. Après quelques rappels sur les codes "cortex", un lien entre ces codes et les treillis cycliques est présenté pour les codes de distance minimale 8, ce qui a notamment permis de trouver un nouveau treillis cyclique pour le code de Golay [24,12,8].

3.1 Présentation

3.1.1 Principe de construction

3.1.1.1 Description formelle

L'une des particularités des codes "cortex" est qu'ils peuvent se construire avec différents codes de base ayant eux-mêmes différents rendements. Dans le cadre de la decription formelle, nous nous plaçons dans le cas où tous les codes de base $\mathcal{C}_b$ sont identiques, de dimension k_b, de rendement $\frac{1}{2}$ sur le corps $\mathbb{F}$ et ayant une matrice génératrice sous forme systématique notée $(I_{k_b}|R_b)$.
On considère la concaténation parallèle de e codes $\mathcal{C}_b$. Le code résultant de cette concaténation possède une matrice génératrice sous forme systématique dont la matrice redondante R_k est donnée par le produit de

Kronecker :

$$R_k = I_e \otimes R_b$$

L'idée des codes "cortex" est d'alterner ces phases de concaténation parallèle avec des phases de permutations adin de mettre en série ces encodages parallèles qui ne permettent pas à eux seuls d'augmenter la distance minimale par rapport au code de base $\mathcal{C}_b$ utilisé.
Avant de définir la construction "cortex" à proprement dite, on introduit quelques notations. On note S_k le groupe symétrique agissant sur un ensemble à k éléments. On peut associer toute permutation π de S_k avec une matrice carrée $\Pi = (h_{i,j})$ de taille $k \times k$ telle que :

$$h_{i,j} = \delta_{\pi(i),j}$$

où

$$\begin{aligned} \forall (i,j) \in \mathbb{N}^* \times \mathbb{N}^*,\ \delta_{i,j} &= 1 \text{ si } i = j \\ &= 0 \text{ sinon} \end{aligned}$$

Soit $m = (m_0, \ldots, m_{k-1})$ un vecteur de $\mathbb{F}^k$. On a la relation :

$$m.\Pi = \left(m_{\pi^{-1}(0)}, \ldots, m_{\pi^{-1}(k-1)}\right)$$

On note $R^{(s)}(m)$ le vecteur de redondance associé à un vecteur m de $\mathbb{F}^k$ ayant subi s permutations alternées avec des encodages parallèles. Ce vecteur est donné par la relation :

$$R^{(s)}(m) = mR_k\Pi_0 R_k \ldots \Pi_{s-1}R_k$$

Définition 3.1.1 *Soit $\mathcal{C}_b$ un code linéaire de dimension k_b, de rendement $\frac{1}{2}$ sur un corps $\mathbb{F}$ et dont une matrice génératrice sous forme systématique se note $(I_{k_b}|R_b)$. Soient $k = ek_b$ où e est un entier non nul et $R_k = I_e \otimes R_b$ (où $\otimes$ définit le produit de Kronecker de 2 matrices). Pour toute suite de permutations $\Pi = (\Pi_0, \ldots, \Pi_{s-1})$ de S_k, le code "cortex" construit à partir du code de base $\mathcal{C}_b$ et utilisant cette suite de permutations est l'ensemble des mots correspondant à la concaténation des vecteurs m et $R^{(s)}(m)$ lorsque m décrit l'espace $\mathbb{F}^{k_b}$.*

Le code ainsi obtenu est un code de dimension k, de rendement $\frac{1}{2}$ et dont une matrice génératrice sous forme systématique est donnée par :

$$G = (I_k|R_b\Pi_0 R_b \ldots \Pi_{s-1}R_b)$$

3.1.1.2 Description graphique

Les codes "cortex" binaires sont des codes correcteurs d'erreurs en blocs systématiques de longueur $n = 2k$ comportant k bits d'information. Ces codes se présentent sous la forme de concaténations série et parallèle de petits codes de base $C_b[n_b, k_b, d_b]$ (la figure 3.1 utilise un code de Hamming [8,4,4] comme code de base noté $\mathcal{H}$). Les codes de base sont regroupés en étages séparés par des permutations (π sur la figure 3.1) non nécessairement identiques.
Le processus d'encodage systématique se déroule de la façon suivante :
Le premier étage transforme le vecteur de bits d'information à encoder $X = (x_0, x_1, ..., x_{k-1}) = (x_0^{(0)}, x_1^{(0)}, ..., x_{k-1}^{(0)}) = X^{(0)}$ en un vecteur de bits de redondance $R^{(0)} = (r_0^{(0)}, r_1^{(0)}, ..., r_{k-1}^{(0)})$. Les éléments de ce vecteur redondance subissent alors une première permutation π_0 pour obtenir $X^{(1)} = (r_{\pi(0)}^{(0)}, r_{\pi(1)}^{(0)}, ..., r_{\pi(k-1)}^{(0)})$. On procède ainsi pour les n_e étages et les $n_e - 1$ permutations qui composent la structure. Le mot de code correspond alors à la concaténation du vecteur X de bits d'information avec le vecteur $R^{(n_e-1)}$ de bits de redondance.

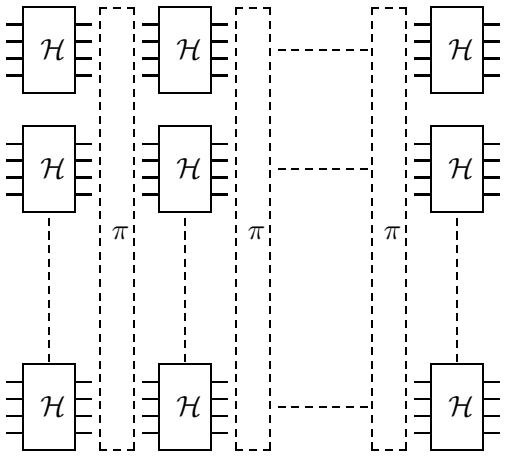


FIG. 3.1 – Codeur "cortex"

3.1.2 Propriétés et codes construits

3.1.2.1 Quelques rappels

En préambule à cette partie, nous rappelons quelques définitions sur les codes binaires auto-duaux. Par convention, n, k et d sont des symboles "réservés" qui désignent respectivement la longueur, la dimension et la distance minimale du code.

Définition 3.1.2 *On note $C^{\perp}$ le code dual du code C défini par :*

$$C^{\perp} = \{y \in \mathbb{F}_2^n / \forall x \in C, x.y = 0\}$$

où "$x \cdot y$" correspond au produit scalaire $\sum_{i=1}^{n} x_i y_i$.

Définition 3.1.3 *Un code est dit auto-orthogonal si $C \subset C^{\perp}$.*

Définition 3.1.4 *Un code est dit auto-dual si $C = C^{\perp}$.*

Remarque 3.1.1 *Les mots d'un code auto-dual sont tous de poids pairs.*

Définition 3.1.5 *Un code auto-dual sur $\mathbb{F}_2$ dont tous les poids sont congrus à $0 \mod 2$ et pour lesquels il existe au moins un mot de poids congru à $2 \mod 4$ est de Type I (i.e simplement pair).*

Définition 3.1.6 *Si tous les mots d'un code binaire C sont de poids congrus à $0 \mod 4$, C est dit de Type II (i.e doublement pair).*

Théorème 3.1.1 *La distance minimale d'un code de type I est bornée par*

$$d \preceq 2 \times \left\lfloor \frac{n}{8} \right\rfloor + 2$$

Théorème 3.1.2 *La distance minimale d'un code de type II est bornée par*

$$d \preceq 4 \times \left\lfloor \frac{n}{24} \right\rfloor + 4$$

Remarque 3.1.2 *Un code dont la distance minimale atteint cette borne est dit extrémal.*

3.1.2.2 Propriétés des codes “cortex”

Les deux théorèmes énoncés, ainsi que leurs démonstrations, proviennent de Canteaut et Charpin [24]. Ces deux théorèmes souligne notamment l'intérêt des codes cortex pour la construction des codes auto-duaux.

Théorème 3.1.3 *Lorsque le nombre de permutations est pair (respectivement impair), les codes "cortex" construits à l'aide du code de Hamming étendu [8,4,4] ont des mots de code de poids multiple de 4 (respectivement de 2).*

Théorème 3.1.4 *Les codes “cortex” construits ayant pour base des codes auto-duaux sont des codes auto-duaux.*

On peut citer deux autres résultats majeurs concernant les codes cortex. Le premier est l'œuvre de Olocco et Tillich [74] qui ont montré que ces codes sont asymptotiquement bon en démontrant qu'ils ont la même capacité de correction que des codes aléatoires.
Un dernier résultat théorique très important provient des travaux de Otmani [77] qui a démontré que tout code auto-dual de type II admet une représentation sous forme de codes cortex.

3.1.2.3 Codes obtenus

Dans cette partie, on récapitule les codes auto-duaux construits par la méthode "cortex" à l'aide de permutation de la forme :

$$i \mapsto (a * i + b) \mod k \text{ pour } (a, b) \in \mathbb{Z} \times \mathbb{Z}.$$

Ces codes sont répertoriés dans le tableau 3.1 et proviennent des travaux de thèse de Otmani [77] à l'exception du [96,48,16] et du [128,64,20] obtenus durant ma thèse.

3.2 Décodage

Les travaux de Olocco et Tillich ont notamment permis de mettre à jour la difficulté de décoder les codes cortex possédants plus de deux permutations. Les différents résultats expérimentaux obtenus durant cette thèse abondent par ailleurs pleinement dans ce sens. Cependant, pour certains codes cortex à deux permutations, notamment ceux de distance minimale 8, il est possible de les décoder en faisant un lien entre ces codes cortex et les

code de paramètres $[n,k,d]$	borne d_{min}	couple (a,b)	nombre d'étages
[16,8,4]	4	(1,0)	3
[24,12,8]	8	(5,1)	3
[32,16,8]	8	(3,0)	3
[40,20,8]	8	(3,0)	3
[56,28,12]	12	(5,1)	3
[64,32,12]	12	(19,0)	3
[72,36,12]	16	(5,0)	3
[88,44,16]	16	(35,0)	7
[96,48,16]	20	(37,1)	21
[128,64,20]	24	(19,0)	21

(en caractères gras les codes auto-duaux extrémaux)

TAB. 3.1 – Codes auto-duaux mis sous forme "cortex"

treillis cycliques. Cette approche a notamment permis de trouver un nouveau treillis cyclique à 16 états pour représenter le code de Golay [24,12,8] (Cf. figure 3.4). Cependant, mis à part certains codes de distance minimale 12 pouvant se mettre sous la forme d'un treillis cyclique à 256 états, le lien entre les codes "cortex" et les treillis cycliques n'est pas une généralité. De plus une telle approche est limitée par l'aspect complexité car plus la distance minimale est importante et plus le treillis cyclique obtenu aura un nombre d'états important, ce qui le rendra trop complexe pour la phase de décodage.

3.2.1 Exemples de codes de distance minimale 8

Dans cette section, nous traitons l'exemple d'un canal de transmission pour lequel le code utilisé est le Golay [24,12,8] associé à une modulation MDP-2. Le bruit est supposé gaussien et additif. Le codeur canal utilisé est le codeur "cortex" de la figure 3.2 pour lequel les boîtes A_i, B_i et C_i sont des codes de Hamming étendu [8,4,4]. Les bits x_i correspondent aux bits d'information alors que les bits r_i sont ceux de redondance. Le décodeur utilise quant à lui une structure dérivée du codeur "cortex" et qui est représentée sur la figure 3.3.
L'obtention de cette nouvelle structure provient directement du code cortex associé mais pour lequel on aurait changé le sens de propagation de l'information. Par exemple, sur la figure 3.2, les bits d'information initialisent la bordure gauche de la structure et la propagation de gauche à droite permet

d'obtenir les bits de redondance sur la bordure droite. Pour obtenir la figure 3.3, il nous suffit de choisir une propagation orthogonale à la précédente en prenant B_0 comme origine. La propagation se fait alors sous forme de 2 ondes qui avancent simultanément et parallèlement. Lorsque l'une de ses ondes traverse une boîte A_i (boîte qui reçoit les bits d'information), l'autre traverse dans le même temps une boîte C_i (boîte qui délivre les bits de redondance) et elles ne se rejoignent que sur les boîtes B_i donnant ainsi naissance à la forme factorisée de la structure.

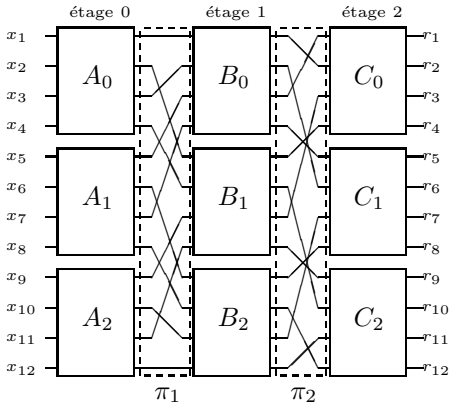


FIG. 3.2 – Codeur "cortex" du code de Golay [24,12,8]

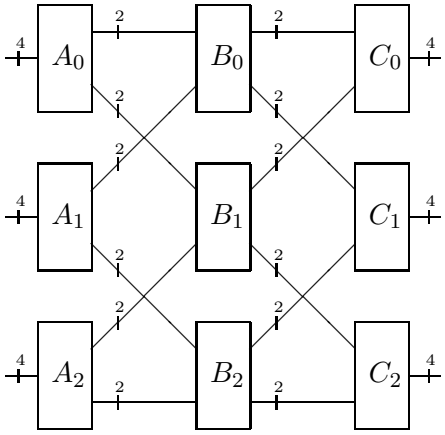


FIG. 3.3 – Codeur "cortex" simplifié du code de Golay [24,12,8]

Le lien important qui existe entre la représentation cortex de la figure 3.2 et le treillis cyclique de la figure 3.4 apparaît plus clairement lorsque l'on regroupe les différentes liaisons entre les codes de Hamming par un seul

et même trait. Ainsi sur la figure 3.3, les liaisons au centre de la structure représentent des couples de bits qui seront des bits d'état pour le treillis cyclique. Le passage de la structure cortex à un treillis cyclique est donc en grande partie dépendant de la permutation utilisée.

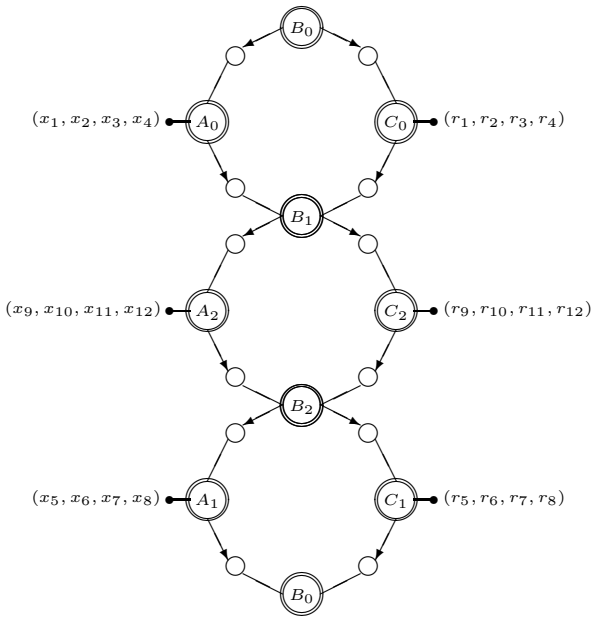


FIG. 3.4 – Code de Golay [24,12,8] sous forme "cortex" dépliée

Le treillis ainsi obtenu (Cf. figure 3.4) présente une structure particulière qui est formée par la concaténation de cycles (appelés "bracelets"). Le code de Golay [24,12,8] présenté à titre d'exemple comporte 3 bracelets correspondant aux cycles (B_0, C_0, B_1, A_0), (B_1, C_2, B_2, A_2) et (B_2, C_1, B_0, A_1). En raison de cette particularité structurelle, ces treillis initialement présentés par Cadic et al. dans [20] seront appelés des "treillis bracelets".
Sur cette nouvelle structure, les différentes boîtes n'ont plus vraiment le même fonctionnement que dans le cas du codeur "cortex" en raison des modifications exercées sur le séquencement, mais elles demeurent en revanche de même nature dans le sens où elles découlent directement de la brique de base de la figure 3.2 à savoir le code de Hamming étendu [8,4,4]. Plus précisément, on déduit deux types de transition que l'on représente sous formes matricielles en utilisant le formalisme suivant : une matrice M de transition des poids d'une section de treillis possédant un ensemble I d'états initiaux et

un ensemble J d'états finaux a ses composantes $m_{i,j}$ telles que :

$$\forall i \in I, \forall j \in J,\ m_{i,j} = Z^{w(br_{i,j})}$$

où Z est une variable formelle et $w(br_{i,j})$ est le poids de Hamming de la branche reliant l'état S_i de début de section à l'état S_j de fin de section.
Lorsque qu'il n'existe pas de branche reliant un état S_i avec un état S_j, on prend comme convention $m_{i,j} = 0$.
On considère maintenant le cas du code de Hamming étendu [8,4,4] pour élaborer nos matrices de transition dont la matrice génératrice est de la forme :

$$G = \begin{pmatrix} 1\,0\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0 \end{pmatrix}$$

- ◇ matrice de transition associée aux boîtes A_i et C_i :
  Ces matrices sont déterminées en considérant que pour chaque mot de code les 4 bits d'information servent à étiqueter la branche du treillis (le poids de ces 4 bits correspond donc à l'exposant de Z) reliant l'état de début (donné par les 2 derniers bits de redondance) à l'état de fin (donné par les 2 premiers bits de redondance). Avec ces conventions nous avons :

$$A_i = C_i = \begin{pmatrix} 1 & Z^3 & Z^3 & Z^2 \\ Z^3 & Z^2 & Z^2 & Z \\ Z^3 & Z^2 & Z^2 & Z \\ Z^2 & Z & Z & Z^4 \end{pmatrix}$$

- ◇ matrice de transition associée aux boîtes B_i :
  Ces matrices sont obtenues en considérant que les 2 premiers bits d'information et les 2 premiers bits de redondance constituent l'état d'entrée de la section alors que les deux derniers bits d'information et de redondandance constituent l'état de sortie de la section. La matrice

ainsi obtenue est la suivante :

$$B_i = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

Ces deux types de matrices sont associés chacun à un treillis bien particulier. Dans le cas A_i ou C_i, cela correspond à un treillis à 4 états possédant 4 bits d'étiquettes et qui est représenté sur la figure 3.5. Par convention, les étiquettes γ notées entre accolades à gauche d'un état S correspondent, lorsqu'elle sont lues de la gauche vers la droite, aux étiquettes reliant cet état S respectivement aux états 0 (00), 1 (01), 2 (10) et 3 (11).
Pour le cas de B_i, il apparaît clairement que c'est une matrice de permutation. Elle est par conséquent associée au treillis de la figure 3.6 qui ne possède aucun bit d'étiquette.

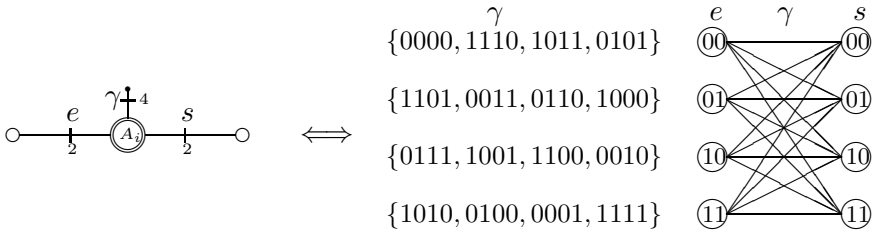


FIG. 3.5 – Treillis associé aux boîtes A_i et C_i

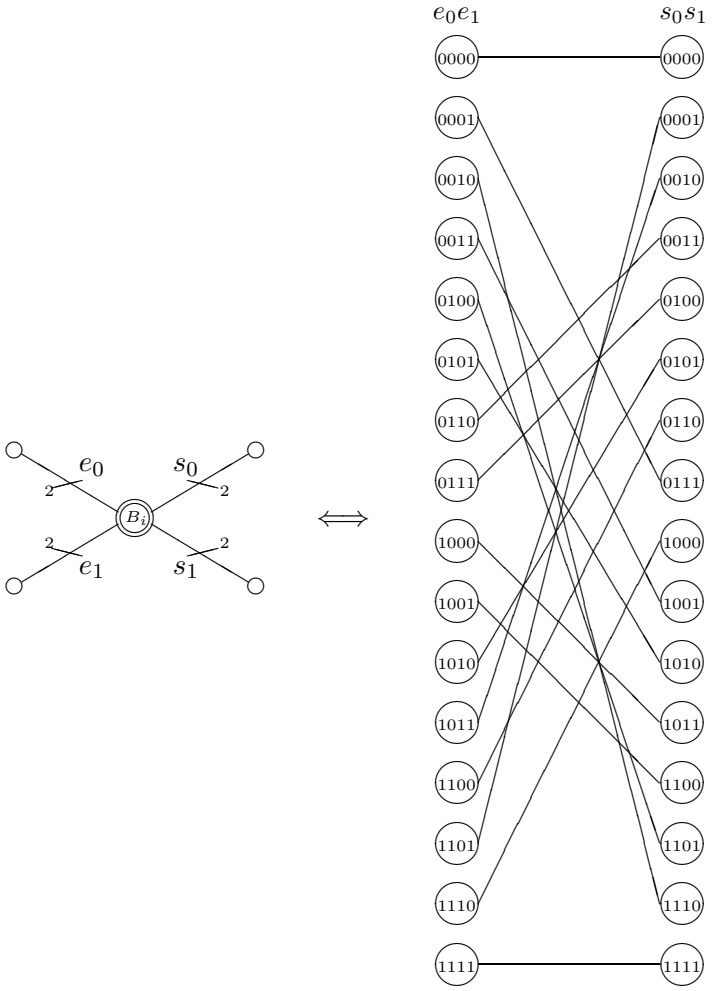


FIG. 3.6 – Treillis associé aux boîtes B_i

3.2.2 Calcul de la distribution des poids

En guise de préambule quelques rappels sur le produit de Kronecker, aussi appelé produit tensoriel.
Soient $A = (a_{i,j})_{\substack{1 \preceq j \preceq n \\ 1 \preceq i \preceq m}}$ et $B = (b_{i,j})_{\substack{1 \preceq j \preceq q \\ 1 \preceq i \preceq p}}$ deux matrices. La matrice C résultant du produit de Kronecker de A et B est la matrice de dimension $mp \times nq$ dont chaque terme est donné par la relation $c_{ij,kl} = a_{i,k}b_{j,l}$. De

façon similaire, C peut s'écrire sous forme partitionnée comme suit :

$$C = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ \vdots & \vdots & & \vdots \\ a_{m,1}B & a_{m,2}B & \dots & a_{m,n}B \end{pmatrix}.$$

L'une des propriétés importantes de ce produit est qu'il est lié avec le produit matriciel classique via la relation :

$$(A \otimes B).(C \otimes D) = (A.C) \otimes (B.D)$$

où A, B, C et D sont des matrices de dimensions repectives $m \times n$, $p \times q$, $n \times r$ et $q \times s$ alors que . et $\otimes$ représentent respectivement le produit matriciel et le produit de Kronecker.
Déterminons maintenant le polynôme de distribution des poids du code de Golay [24,12,8] par cette méthode. Ce treillis du code de Golay [24,12,8] (Cf. figure 3.4) a la particularité d'être une succession de sections, reliées entre elles par d'autres sections agissant comme des permutations, sur lesquelles le treillis est factorisé en deux sous-treillis identiques. Ainsi, pour représenter formellement cette factorisation, on utilise le produit de Kronecker qui réalise alors le passage de deux espaces de dimension quatre à un espace de dimension seize. En d'autres termes le produit de Kronecker de deux matrices associées chacune à une section de treillis correspond au produit de ces deux sections de treillis. Par suite, le polynôme énumérateur des poids $W(Z)$ du code de Golay [24,12,8] est donné par la relation :

$$W(Z) = Trace[\prod_{i=0}^{2}(A_{(2i)mod3} \otimes C_{(2i)mod3}).B_{(2i+1)mod3}]$$

En appliquant cette formule avec les matrices $(A_i)_{i=0\dots2}$, $(B_i)_{i=0\dots2}$ et $(C_i)_{i=0\dots2}$ définies précédemment, on obtient comme polynôme énumérateur des poids :

$$W(Z) = 1 + 759Z^8 + 2576Z^{12} + 759Z^{16} + Z^{24}$$

De plus quelque soit le nombre de bracelets que l'on rajoute au treillis, la distance minimale demeure inchangée, à savoir 8 [20].
Être en mesure de calculer le polynôme énumérateur des poids d'un code à partir d'une structure graphique est très important car, d'après Aji et McEliece [2], cela assure que l'algorithme de décodage adapté de ce calcul de

distribution sera optimal selon le critère du maximum de vraisemblance. Il est cependant très difficile, voire impossible, de déterminer de façon exacte par un calcul simple le polynôme énumérateur des poids d'un code sur des structures autres que les arbres ou les treillis. L'algorithme de décodage décrit dans la section 3.2.3. fonctionne suivant le modèle de calcul de la distribution des poids présenté dans cette section et est par conséquent optimal selon le critère du maximum de vraisemblance.

3.2.3 Justification de la méthode de décodage

Avant de définir quelques notations, nous rappelons que le décodage s'effectue à partir d'un treillis cyclique comportant n sections (identiques dans le cadre de l'exemple) dont l'une d'elles est schématisée sur la figure 3.7. On rappelle également qu'une classe d'équivalence correspond à l'ensemble des mots de code dont la représentation sur le treillis est un circuit ou chemin fermé dont l'état de départ est identique à celui d'arrivée.
La figure 3.7 représente une seule section du treillis avec 2 branches parallèles qui correspondent aux sous-treillis A_t et C_t comportant respectivement M_A et M_C états alors que B_t et B_{t+1} sont des treillis d'interaction sans étiquette possédant $M = M_A M_C$ états.

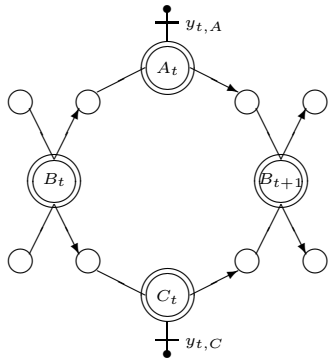


FIG. 3.7 – Schéma d'une section factorisée (bracelet) du treillis

On note $X_i = (x_0, x_1, \ldots, x_{n-1}) \in \mathcal{C}$ un mot de code et $Y = (y_0, y_1, \ldots, y_{n-1})$ une observation reçue. En fait, chacun des c_t correspond à l'ensemble des bits que l'on rencontre sur la section t du treillis : une première moitié de ces bits sert à étiqueter le sous-treillis A_t (Cf. figure 3.7) et l'autre moitié à étiqueter le sous-treillis C_t. De la même façon que l'on répartit les bits d'étiquettes émis sur les deux sous-treillis A_t et B_t, chaque y_t correspond aux observations reçues pour l'ensemble des bits de la section

t. On décompose donc y_t en $y_{t,A}$ et $y_{t,C}$ qui sont les observations reçues permettant de déterminer les métriques de branche respectivement de A_t et C_t.
On note également Y_0^t la suite $(y_0, ..., y_t)$, la suite d'états s_0 à s_t étant de même notée S_0^t.
De plus l'ensemble des probabilités d'atteindre à l'instant t l'état m pour tous les états de départ $0, 1, ..., M-1$ est représenté par la liste :

$$L_t(m) = \begin{pmatrix} Pr(Y_0^t|s_0 = 0) \\ Pr(Y_0^t|s_0 = 1) \\ \vdots \\ Pr(Y_0^t|s_0 = M-1) \end{pmatrix}$$

Avec ces notations, nous pouvons écrire que le décodage d'un code $\mathcal{C}$ selon le maximum de vraisemblance a posteriori consiste à choisir le mot de code $X_i \in \mathcal{C}$ qui maximise la probabilité d'avoir reçu l'observation Y. De façon plus formelle, cela signifie résoudre le problème de maximisation :

$$\max_{X_i \in \mathcal{C}} P(Y|X_i)$$

Or, un mot de code est décrit par un ensemble de transitions entre états s_t et s_{t+1} de deux étages successifs d'indices t et $(t+1)$ d'un treillis. Ce processus de parcours du treillis étant de type Markovien d'ordre 1 (un état s_{t+1} à l'instant $t+1$ ne dépend que de l'état s_t à l'instant t et de la transition), si l'on émet l'hypothèse que les échantillons bruités sont indépendants, nous pouvons écrire :

$$Pr(Y_0^{n-1}|X_i) = \prod_{t=0}^{n-1} [Pr(y_{t,A}|s_t, s_{t+1}) \times Pr(y_{t,C}|s_t, s_{t+1})]$$

où $P(y_{t,A}|s_t, s_{t+1})$ (resp. $P(y_{t,C}|s_t, s_{t+1})$) est la probabilité a priori de la branche du sous-treillis A_t (resp. C_t) associée à la transition entre les états s_t et s_{t+1}.
De plus, on impose $s_0 = s_{n-1}$. En effet, le treillis étant cyclique, l'état de départ et celui d'arrivée doivent être les mêmes. Notre problème de maximisation de la vraisemblance a posteriori $Pr(Y_0^{n-1}|X_i)$ consiste donc à rechercher quelque soit m variant de 0 à $M-1$ la valeur optimale :

$$\max_{s_0 = s_{n-1} = m} \prod_{t=0}^{n-1} [Pr(y_{t,A}|s_t, s_{t+1}) \times Pr(y_{t,C}|s_t, s_{t+1})]$$

Justifions maintenant, à l'aide d'un raisonnement par récurrence en faisant varier de 0 à n le nombre de sections factorisées (bracelets) de notre treillis cyclique, le fait que notre méthode permet bien la résolution d'un tel problème de maximisation.

1. Initialisation ($t = 0$).
   Pour chacun des $s_0(m)$, m variant de 0 à $M-1$, les probabilités sont initialisées comme suit :

$$\begin{cases} Pr(Y_0^0|s_0 = i) = 1/M & \text{si m = i} \\ Pr(Y_0^0|s_0 = i) = 0 & \text{sinon} \end{cases} \quad \text{pour } i \text{ variant de 0 à } M-1.$$

2. Treillis cyclique comportant 1 section factorisée ($t = 1$).
   Pout tout m variant de 0 à $M-1$, on obtient :

$$L_1(m) = \begin{pmatrix} \max\limits_{S_0^1} [Pr(y_{0,A}|s_0 = 0) \times Pr(y_{0,C}|s_0 = 0)] \\ \vdots \\ \max\limits_{S_0^1} [Pr(y_{0,A}|s_0 = M-1) \times Pr(y_{0,C}|s_0 = M-1)] \end{pmatrix}$$

   Pour obtenir la séquence optimale S_0^1, il nous faut alors prendre :

$$\max_m L_1(m) = \max_{S_0^1, m} [Pr(y_{0,A}|s_0 = s_1 = m) \times Pr(y_{0,C}|s_0 = s_1 = m)]$$

3. Treillis cyclique comportant t section factorisées.
   A un instant donné t, on dispose de l'ensemble des vecteurs optimaux définis par :

$$L_t(m) = \begin{pmatrix} Pr(Y_0^t|s_0 = 0) \\ \vdots \\ Pr(Y_0^t|s_0 = M-1) \end{pmatrix}, \text{ pour } m \text{ variant de 0 à } M-1.$$

   Par suite, à l'instant $t+1$, nous obtenons :

$$L_{t+1}(m) = \begin{pmatrix} \max\limits_{S_0^{t+1}} Pr(Y_0^{t+1}|s_0 = 0) \\ \vdots \\ \max\limits_{S_0^{t+1}} Pr(Y_0^{t+1}|s_0 = M-1) \end{pmatrix}, \text{ pour } m \text{ variant de 0 à } M-1.$$

avec :

$$\max_{S_0^{t+1}} Pr(Y_0^{t+1}|s_0) = \max_{S_0^t} \left(Pr(Y_0^t|s_0) \times \{ Pr(y_{t+1,A}|s_t, s_{t+1}) \times Pr(y_{t+1,C}|s_t, s_{t+1}) \} \right)$$

La séquence optimale S_0^{t+1} est alors obtenue à l'aide de la relation :

$$\max_m L_{t+1}(m) = \max_{S_0^{t+1},m} Pr(Y_0^{t+1}|s_0 = s_{t+1} = m)$$

Par récurrence, nous pouvons donc généraliser ce résultat à chacune de nos itérations et en déduire que le chemin obtenu au terme de l'iteration $(n-1)$ correspond à un mot de code valide et optimal pour la résolution de notre problème.

La figure 3.8 présente les différentes courbes de taux d'erreur binaire binaire obtenues avec cette méthode selon que l'on conserve ou non l'intégralité des classes d'équivalence tout au long du processus. Ainsi, la courbe nommée "16 classes" signifie que toutes les classes de départ sont préservées. Plus généralement, les courbes "n classes" correspondent à un décodage pour lequel on ne conserve que les n classes de plus fortes vraisemblances en chacun des états du treillis.

La figure 3.9 présente les courbes de décodage obtenues pour un code paramètres [256,128,8] associé lui aussi à un treillis cyclique de type "bracelet".

3.2.4 Complexité

Les résultats des diverses simulations viennent confirmer le fait que le décodage est optimal lorsque que l'on conserve l'intégralité des classes d'équivalence tout au long du processus. De plus, les simulations laissent entrevoir le fait qu'il peut être intéressant de ne pas conserver toutes les classes d'équivalence permettant ainsi de limiter la complexité. Par exemple, si l'on conserve seulement 8 classes, alors les résultats sont très proches de l'optimal (écart inférieur à 0.1 dB). Si l'on ne conserve que 4 classes, la courbe s'écarte légèrement de l'optimal (environ 0.1 dB). En revanche, avec seulement deux classes d'équivalence, la courbe s'éloigne nettement de l'optimal (environ 0.8 dB).
Un bon compromis semble donc être le décodage conservant 4 classes d'équivalence car s'il permet de réduire la complexité en divisant par 4 le nombre de classes d'équivalence, il parvient tout de même à demeurer quasi optimal en terme de correction d'erreurs.
Les deux tableaux 3.2 et 3.3 sont relatifs à l'étude de la complexité de

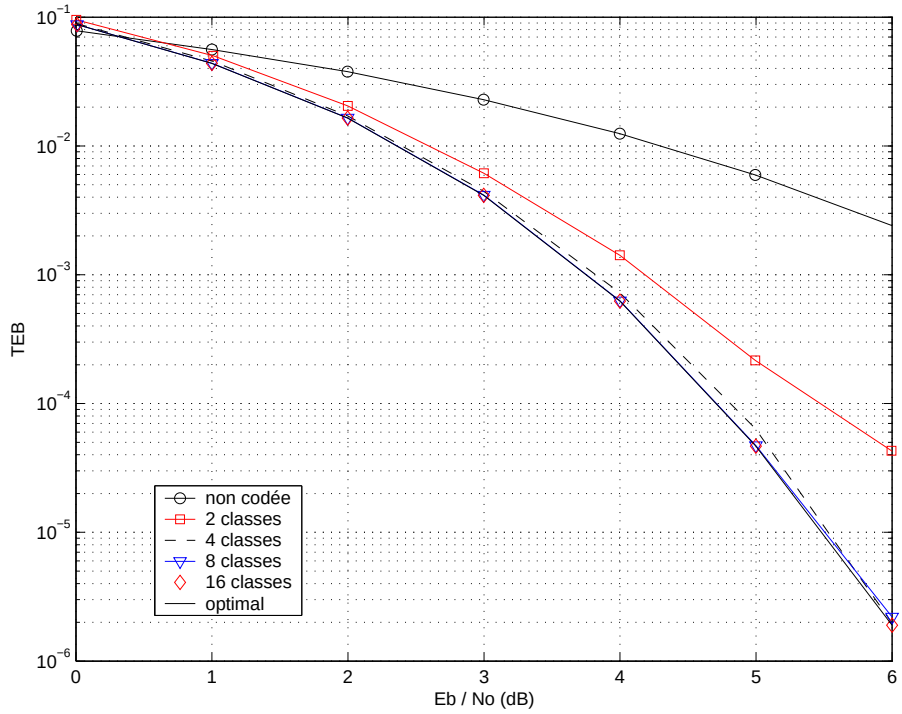


FIG. 3.8 – Décodage du code de Golay [24,12,8]

notre algorithme sur différents treillis du code de Golay [24,12,8]. Dans le premier ce sont les nombres d'additions nécessaires au fonctionnement du programme qui sont recensés alors que le second exprime des gains de complexité entre les différents treillis. De ces résultats, il se dégage le fait que le treillis cyclique factorisé est le plus performant en terme de complexité et son avantage devient de plus en plus évident au fur et à mesure que l'on diminue le nombre de classes survivantes. La notation treillis cyclique "CFV" fait référence aux initiales des auteurs Calderbank et al. de [21] dans lequel un treillis minimal cyclique pour le code de Golay est présenté. Ce treillis possède 12 sections à 16 états comportant chacune 2 bits (l'un d'information et l'autre de redondance). De plus, pour chaque section, chaque état possède deux branches entrantes et deux branches sortantes. Le gain de complexité de notre treillis par rapport au treillis "CFV" provient ainsi en grande partie du calcul des métriques sur les branches du treillis. Dans les cas du treillis bracelet, il y a 32 branches de 4 bits par nœud de bracelet et donc $3 \times 32 = 96$ branches de 4 bits à traiter pour le code de Golay [24,12,8]. Dans le cas du

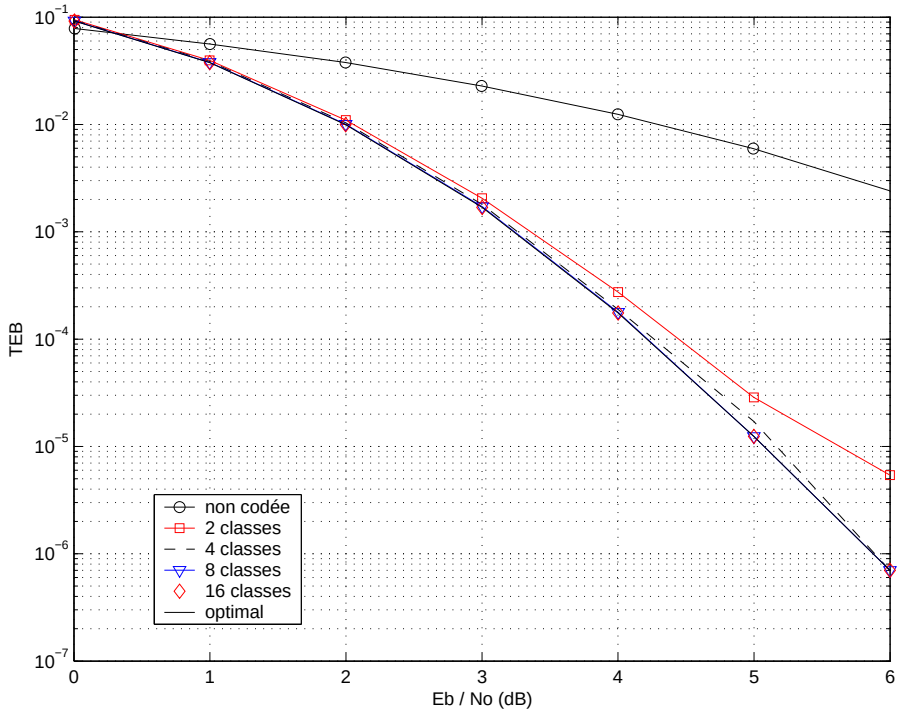


FIG. 3.9 – Décodage du code de paramètres [256,128,8]

treillis "CFV", il y a $16 \times 2 = 32$ branches de 2 bits pour chaque section. Par conséquent pour ce même code de Golay, il faut traiter $32 \times 12 = 384$ branches de 2 bits.

Remarque 3.2.1 *On peut constater que la complexité de décodage est fortement liée au nombre de classes conservées car l'algorithme de décodage utilisé pour chacun des treillis est celui présenté en 3.2.3.*

Remarque 3.2.2 *Les travaux réalisés concernant le décodage des codes cortex binaires construits avec deux permutations et de distance minimale 8 peuvent être appliqués aux codes cortex sur $\mathbb{Z}_4$ (l'Octacode [8,4,6] remplace alors le Hamming [8,4,4]) composés de deux permutations et de distance minimale 12. En effet, dans ce cas, la structure cortex peut là aussi se déplier sous la forme d'un treillis "bracelet" comportant cette fois-ci 256 états.*

nombre de classes conservées	nombre d'additions dans l'algorithme de décodage		
	treillis cyclique "CFV"	treillis cyclique non-factorisé	treillis cyclique factorisé
16 classes	5092	2301	1626
8 classes	2788	1533	858
4 classes	1636	1149	474
2 classes	1060	957	282

TAB. 3.2 – Complexité de décodage par "classes"

nombre de classes conservées	gain de complexité	
	"CFV" / factorisé	non-factorisé / factorisé
16 classes	3.13	1.41
8 classes	3.25	1.79
4 classes	3.45	2.42
2 classes	3.76	3.39

TAB. 3.3 – Comparaison de la complexité de décodage

3.3 Codes à information répartie

3.3.1 Motivation

L'un des enseignements importants de l'étude des codes "cortex" de Carlach et al. [25] est la difficulté de décoder des codes construits à partir de plus de trois couches successives composées de petits codes de base (ici le Hamming étendu [8,4,4]). En effet, ces couches de petits codes ne sont pas en relation directe avec les bits reçus par le canal de transmission, ce qui induit une "dilution" de l'information au fur et à mesure de la progression dans la structure. Ce phénomène est d'autant plus important que les couches de codes sont nombreuses.
Une des solutions envisagées durant cette thèse pour lutter contre ce phénomène est une évolution de la construction "cortex" vers une structure offrant une meilleure répartition des bits destinés à être émis sur le canal. Cela signifie que les bits composant un mot de code ne doivent plus setrouver uniquement sur les bords de la structure comme cela est le cas avec les codes "cortex", mais au contraire des bits d'information et/ou de redondance doivent apparaître sur chacun des étages de la structure. Ainsi, lors de

l'initialisation du décodage, chacun de ces étages possèdera de l'information provenant directement du canal.

3.3.2 Construction de codes de rendement $1/2$

3.3.2.1 Construction avec un unique code de base

Les codes construits dans cette section ont pour code de base le Hamming étendu [8,4,4] de matrice génératrice

$$G = \begin{pmatrix} 1\,0\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0 \end{pmatrix}$$

L'encodage de l'information se fait progressivement jusqu'au centre de la structure depuis la partie gauche (Cf. figure 3.10), puis sur la partie droite de la structure on recueille la redondance. Par exemple, sur la figure 3.10, les seize premiers bits rencontrés lors du parcours de gauche à droite de la structure correspondent aux bits d'information et les seize derniers aux bits de redondance.
En prenant pour permutation

$$\begin{array}{rccc} & \{0,1,\ldots,15\} & \longrightarrow & \{0,1,\ldots,15\} \\ \pi : & i & \longmapsto & (7 \times i + 1) \mod 16 \end{array}$$

on obtient le code [32,16,6] de la figure 3.10 qui est composé de six couches.

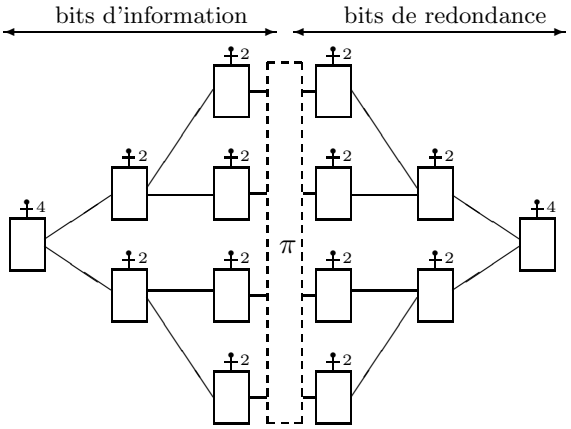


FIG. 3.10 – Code auto-dual de paramètres [32,16,6]

Le polynôme énumérateur des poids de ce code est :

$$\begin{aligned} W(X) &= 1 + 32X^6 + 300X^8 + 1952X^{10} + 6976X^{12} + 14400X^{14} + 18214X^{16} \\ &\quad + 14400X^{18} + 6976X^{20} + 1952X^{22} + 300X^{24} + 32X^{26} + X^{32} \end{aligned}$$

Les deux codes qui suivent sont deux autres exemples de la mise en application de ce principe de construction.

- ◇ Un code de paramètres [64,32,10] obtenu en 8 couches avec les permutations suivantes :

$$\begin{cases} \pi_0, \pi_6 & : & i & \longmapsto & i & \mod 4 \\ \pi_1, \pi_4 & : & i & \longmapsto & (3i+1) & \mod 8 \\ \pi_2, \pi_5 & : & i & \longmapsto & (5i+1) & \mod 16 \\ \pi_3 & : & i & \longmapsto & 19i & \mod 32 \end{cases}$$

- ◇ Un code de paramètres [128,64,14] obtenu en 10 couches avec les permutations suivantes :

$$\begin{cases} \pi_0, \pi_8 & : & i & \longmapsto & i & \mod 4 \\ \pi_1, \pi_7 & : & i & \longmapsto & i & \mod 8 \\ \pi_2, \pi_6 & : & i & \longmapsto & (3i+1) & \mod 16 \\ \pi_3, \pi_5 & : & i & \longmapsto & (19i+1) & \mod 32 \\ \pi_4 & : & i & \longmapsto & (37i+1) & \mod 64 \end{cases}$$

On peut mentionner deux remarques d'ordre expérimental.

Remarque 3.3.1 *La migration sur $\mathbb{Z}_4$ de la structure en utilisant l'octacode [8,4,6] peut s'avérer intéressante si l'on se réfère aux résultats obtenus avec la construction "cortex".*

Remarque 3.3.2 *La distance minimale du code construit semble directement liée à la distance minimale du code pris comme brique de base, car de petites manipulations prenant comme autre brique de base un code [32,16,8] ont permis un accroissement plus rapide de la distance minimale du code global (cf. section suivante).*

3.3.2.2 Construction avec deux codes de base distincts

Ici les deux codes de base utilisés sont le code de Hamming étendu [8,4,4] rencontré précédemment et le code auto-dual [32,16,8] de matrice génératrice

$G = (Id_{16}|P)$ avec

$$P = \begin{pmatrix}
1&1&0&0&0&1&1&1&1&0&1&1&0&1&1&1\\
0&1&1&1&1&1&0&0&0&1&1&1&1&0&1&1\\
1&0&1&1&0&1&1&1&1&1&0&0&0&1&1&1\\
0&1&1&1&1&1&0&1&1&0&1&1&1&1&1&0&0\\
1&1&0&0&1&0&1&1&0&1&1&1&1&0&1&1\\
1&0&1&1&1&1&1&0&0&1&0&1&1&0&1&1&1\\
0&1&1&1&1&1&0&1&1&1&1&0&0&1&0&1&1\\
1&0&1&1&0&1&1&1&1&0&1&1&1&1&0&0\\
0&0&1&1&1&1&1&0&1&1&1&1&0&1&1&0&1\\
1&1&0&1&0&0&1&1&1&1&0&1&1&1&1&0\\
1&1&1&0&1&1&0&1&0&0&1&1&1&1&0&1\\
1&1&0&1&1&1&1&1&0&1&1&0&1&0&0&1&1\\
0&0&1&1&1&1&1&1&0&1&1&0&1&1&1&1&0\\
1&1&1&0&0&0&1&1&1&1&1&1&0&1&1&0&1\\
1&1&0&1&1&1&1&1&0&0&0&1&1&1&1&1&0\\
1&1&1&0&1&1&0&1&1&1&1&1&0&0&0&1&1
\end{pmatrix}$$

Ce code possède de plus la particularité d'être obtenu par la méthode "cortex" avec une architecture à trois couches utilisant le Hamming étendu comme brique de base et la permutation "$i \longmapsto 3i \mod 16$" pour lier ces différentes couches.
Le principe de construction et donc la structure générale restent inchangés, mais on remplace les deux couches de codes de Hamming du centre par deux couches de codes [32,16,8] mentionnés ci-dessus.
En utilisant cette variante de la construction, nous avons construit deux nouveaux codes.

- ◇ Un code de paramètres [128,64,16] obtenu en 10 couches avec les permutations suivantes :

$$\begin{cases}
\pi_0, \pi_8 & : & i & \longmapsto & i & \mod 4\\
\pi_1, \pi_7 & : & i & \longmapsto & i & \mod 8\\
\pi_2, \pi_6 & : & i & \longmapsto & 3i & \mod 16\\
\pi_3, \pi_5 & : & i & \longmapsto & 19i & \mod 32\\
\pi_4 & : & i & \longmapsto & 37i & \mod 64
\end{cases}$$

- ◇ Un code de paramètres [256,128,$d_{min} \succeq 22$] obtenu en 12 couches avec

les permutations suivantes :

$$\begin{cases} \pi_0, \pi_{10} : i \longmapsto i \mod 4 \\ \pi_1, \pi_9 : i \longmapsto i \mod 8 \\ \pi_2, \pi_8 : i \longmapsto 3i \mod 16 \\ \pi_3, \pi_7 : i \longmapsto 19i \mod 32 \\ \pi_4, \pi_6 : i \longmapsto 37i \mod 64 \\ \pi_5 : i \longmapsto 73i \mod 128 \end{cases}$$

3.3.3 Preuve de l'auto-dualité des codes construits

En préambule à la démonstration, nous faisons deux remarques.

Remarque 3.3.3 *La concaténation de codes auto-duaux engendre un code également auto-dual.*

Remarque 3.3.4 *Soient π une permutation, v_0 et v_1 deux vecteurs binaires. Alors,*

$$< v_0, v_1 > \equiv < \pi(v_0), \pi(v_1) > \mod 2$$

Soient v_0 et v_1 deux vecteurs binaires, on note $v = (v_0|v_1)$ le vecteur résultant de la concaténation de v_0 et v_1.
On réalise la démonstration en prenant comme support le cas d'une structure possédant quatre couches de codes (Cf. figure 3.11).
De plus, il est aisé de constater que cette démonstration reste valide quelque soit le nombre de couches composant le code global par le simple fait que ce critère n'intervient pas sur le principe même de la démonstration.
Il est également à noter que cette démonstration demeure vraie dans le cas des deux variantes de la construction (sections 3.3.2.1 et 3.3.2.2) car les codes utilisés (un Hamming étendu [8,4,4] et un code [32,16,8]) sont auto-duaux.

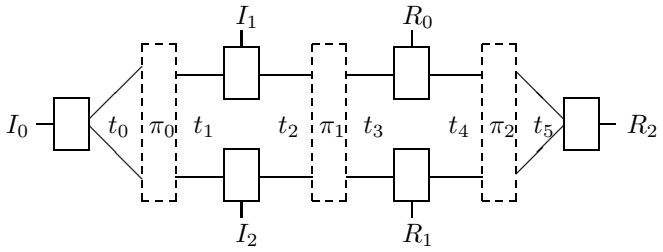


FIG. 3.11 – Codeur à information répartie

Soient $M = (I_0|I_1|I_2|R_0|R_1|R_2)$ et $M' = (I'_0|I'_1|I'_2|R'_0|R'_1|R'_2)$ deux mots quelconque du code C représenté sur la figure 3.11. On note respectivement $t_0, t_1, ..., t_5$ et $t'_0, t'_1, ..., t'_5$ les vecteurs binaires intermédiaires (c'est à dire présents sur la structure, mais n'appartenant pas au mot de code) de ces deux mots M et M'.
Avec ces notations, montrer que le code C est auto-dual revient alors à montrer que l'on a

$$< M, M' > \equiv 0 \mod 2$$

Le code de base étant auto-dual, en utilisant la remarque 3.3.3, nous pouvons écrire les quatres relations de congruences suivantes :

$$< (I_0|t_0), (I'_0|t'_0) > \equiv 0 \mod 2 \tag{3.1}$$
$$< (I_1|t_1|I_2|t_2), (I'_1|t'_1|I'_2|t'_2) > \equiv 0 \mod 2 \tag{3.2}$$
$$< (t_3|R_0|t_4|R_1), (t'_3|R'_0|t'_4|R'_1) > \equiv 0 \mod 2 \tag{3.3}$$
$$< (t_5|R_2), (t'_5|R'_2) > \equiv 0 \mod 2 \tag{3.4}$$

D'après la remarque 3.3.4, nous avons

$$< t_0, t'_0 > \equiv < \pi_0(t_0), \pi_0(t'_0) > \mod 2$$

Par conséquent :

$$< t_0, t'_0 > \equiv < t_1, t'_1 > \mod 2 \tag{3.5}$$

De façon similaire, nous obtenons les deux relations suivantes :

$$< t_4, t'_4 > \equiv < t_5, t'_5 > \mod 2 \tag{3.6}$$
$$< t_2, t'_2 > \equiv < t_3, t'_3 > \mod 2 \tag{3.7}$$

D'après (3.1), nous avons

$$< I_0, I'_0 > \equiv < t_0, t'_0 > \mod 2$$

En utilisant (3.5), il vient alors :

$$< I_0, I'_0 > \equiv < t_1, t'_1 > \mod 2 \tag{3.8}$$

De même, des relations (3.4) et (3.6) nous déduisons :

$$< R_2, R'_2 > \equiv < t_4, t'_4 > \mod 2 \tag{3.9}$$

En combinant (3.2) et (3.8), on obtient :

$$< I_0, I'_0 > + < I_1, I'_1 > + < I_2, I'_2 > + < t_2, t'_2 > \equiv 0 \mod 2$$

De façon identique, la combinaison de (3.3) et (3.9) donne :

$$< R_0, R'_0 > + < R_1, R'_1 > + < R_2, R'_2 > + < t_3, t'_3 > \equiv 0 \mod 2$$

Finalement, de ces deux dernières relations de congruence et de (3.7), nous pouvons déduire :

$$\begin{aligned} &< I_0, I'_0 > + < I_1, I'_1 > + < I_2, I'_2 > + < R_0, R'_0 > \\ &\qquad + < R_1, R'_1 > + < R_2, R'_2 > \qquad \equiv 0 \mod 2 \end{aligned}$$

Autrement dit,

$$< (I_0|I_1|I_2|R_0|R_1|R_2), (I'_0|I'_1|I'_2|R'_0|R'_1|R'_2) > \equiv 0 \mod 2$$

Nous avons donc montré que pour tous mots M et M' appartenant au code C, la relation

$$< M, M' > \equiv 0 \mod 2$$

est vérifiée.
De plus, les codes construits étant de rendement $\frac{1}{2}$, cela signifie que le code et son dual sont de même dimension et possèdent donc le même nombre de mots. De ce fait, il ne peut y avoir d'autres mots que ceux de C pouvant être orthogonaux à chacun des mots de C.
Le code C est donc bien un code auto-dual.

3.3.4 Construction de codes de rendement $2/3$

3.3.4.1 Principe et exemples

Dans cette section, on augmente le rendement du code en couplant par deux les bits de redondance obtenus pour un code de rendement $\frac{1}{2}$ puis en effectuant leur addition binaire (Cf. figure 3.12). Ainsi, on divise par deux le nombre de bits de redondance pour finalement obtenir un code de rendement $\frac{2}{3}$. L'exemple de la figure 3.12 reprend l'architecture du [32,16,6] (cf. section 3.3.2.1) et correspond à un code [24,16,4] de polynôme énumérateur des poids

$$\begin{aligned} W(X) &= 1 + 80X^4 + 1060X^6 + 5746X^8 + 15332X^{10} + 21056X^{12} \\ &\quad +15436X^{14} + 5669X^{16} + 1068X^{18} + 88X^{20}. \end{aligned}$$

Les permutations utilisées pour ce code sont :

$$\begin{cases} \pi_0 : & i & \longmapsto & 3 \times i & \mod 4 \\ \pi_1 : & i & \longmapsto & 3 \times i & \mod 8 \\ \pi_2 : & i & \longmapsto & 3 \times i & \mod 4 \end{cases}$$

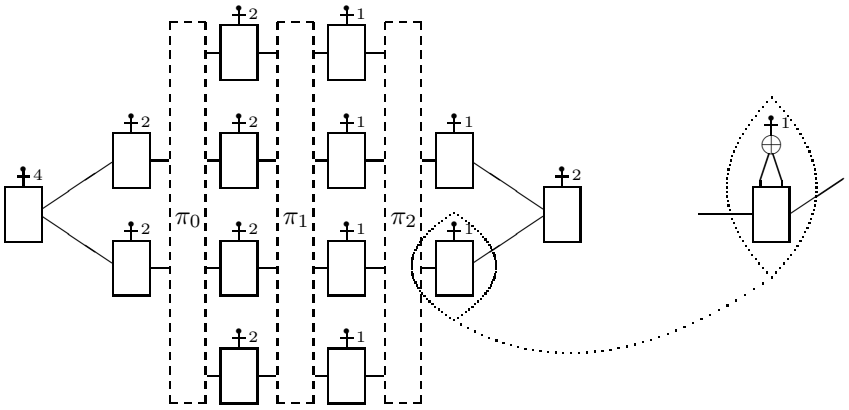


FIG. 3.12 – Code de paramètres [24,16,4]

Remarque 3.3.5 *Le code construit ci-dessus ne possède que des mots de poids pair. Cela reste vrai pour les deux codes mentionnés ci-dessous. Plus généralement, on peut dire que les codes de rendement $\frac{2}{3}$ construits n'ont que des mots de poids pair (cf. section 3.3.4.2).*

En utilisant l'architecture de la section 3.3.2 et en appliquant la modification proposée ci-dessus, nous avons construit les deux codes suivants :

- ◇ Un code de paramètres [96,64,8] obtenu en 10 couches avec les permutations suivantes :

$$\begin{cases} \pi_0, \pi_8 & : & i & \longmapsto & i & \mod 4 \\ \pi_1, \pi_7 & : & i & \longmapsto & i & \mod 8 \\ \pi_2, \pi_6 & : & i & \longmapsto & 3i & \mod 16 \\ \pi_3, \pi_5 & : & i & \longmapsto & 7i & \mod 32 \\ \pi_4 & : & i & \longmapsto & 37i & \mod 64 \end{cases}$$

- ◇ Un code de paramètres [192,128,14] obtenu en 12 couches avec les

permutations suivantes :

$$\begin{cases} \pi_0, \pi_{10} & : & i & \longmapsto & i & \mod 4 \\ \pi_1, \pi_9 & : & i & \longmapsto & i & \mod 8 \\ \pi_2, \pi_8 & : & i & \longmapsto & 3i & \mod 16 \\ \pi_3, \pi_7 & : & i & \longmapsto & 19i & \mod 32 \\ \pi_4, \pi_6 & : & i & \longmapsto & 37i & \mod 64 \\ \pi_5 & : & i & \longmapsto & 73i & \mod 128 \end{cases}$$

3.3.4.2 Parité du poids de Hamming des mots de code obtenus

Il n'est pas difficile de constater que les codes construits présentent nécessairement la particularité de n'avoir que des mots de poids pair.
En effet, dans la section 3.3.3.2 nous avons montré que les codes de rendement $\frac{1}{2}$ obtenus sont auto-duaux et que par conséquent ils ne possèdent que des mots de poids pair. Pour obtenir un code de rendement $\frac{2}{3}$ nous couplons simplement les bits de redondance avant d'en faire l'addition binaire. De ce fait pour assurer que le mot obtenu après addition binaire est de poids pair, il suffit de vérifier que le poids de la redondance avant addition (noté $W(R)$) est identique au poids de la nouvelle redondance obtenue après addition (noté $W(R')$).
Pour cela, on peut énumérer les quatre cas de figure possibles :

(i) la redondance du code de rendement $\frac{1}{2}$ possède un nombre pair de couples (1,1) et un nombre pair de couples (1,0), ce qui signifie que $W(R)$ est pair. Après addition binaire, un couple (1,1) donne un poids 0 et un couple (1,0) un poids 1. Par conséquent après avoir effectué les additions sur chacun des couples, on obtient un nombre pair de 1. Donc $W(R')$ est pair.

(ii) un nombre pair de couples (1,1) et un nombre impair de couples (1,0) impliquent que $W(R)$ est impair. Après addition binaire sur les couples, on obtient un nombre impair de 1 et donc $W(R')$ est lui aussi impair.

(iii) un nombre impair de couples (1,1) et un nombre pair de couples (1,0) impliquent que $W(R)$ est pair. Après addition binaire sur les couples, on obtient un nombre pair de 1 et donc $W(R')$ est lui aussi pair.

(iv) un nombre impair de couples (1,1) et un nombre impair de couples (1,0) impliquent que $W(R)$ est impair. Après addition binaire sur les

couples, on obtient un nombre impair de 1 et donc $W(R')$ est lui aussi impair.

Par conséquent, l'architecture proposée conduit nécessairement à l'obtention de codes de rendement $\frac{2}{3}$ dont les mots sont tous de poids pair.

3.3.5 Lien avec les codes cortex

Dans cette partie nous mettons en avant la relation qu'il existe entre les "codes cortex" et les codes à treillis factorisé en montrant que tout "code cortex" peut s'écrire sous la forme d'un code à treillis factorisé possédant une unique boîte de Hamming comme racine. Dans le cas présent, une structure "cortex" est équivalente à une structure factorisée possédant une boîte de Hamming comme racine et pour laquelle on alterne les étages portant des étiquettes avec des étages sans étiquette. Par exemple, la figure 3.15 permet d'obtenir un code auto-dual de paramètres [64,32,12] identique à celui obtenu par la construction cortex. Il en va de même pour les figures 3.13 et 3.14.

Remarque 3.3.6 *Montrer que tout code cortex peut se mettre sous cette forme est un résultat trivial lorsque l'on remarque que toutes les connections de la structure, exceptées les deux permutations π, sont en fait des permutations réalisant l'identité.*

Les espoirs placés dans de tels codes permettant de répartir l'information sur la structure ne se sont malheureusement pas concrétisés et les différentes expérimentation concernant le décodage se sont révélées extrêmement décevantes.

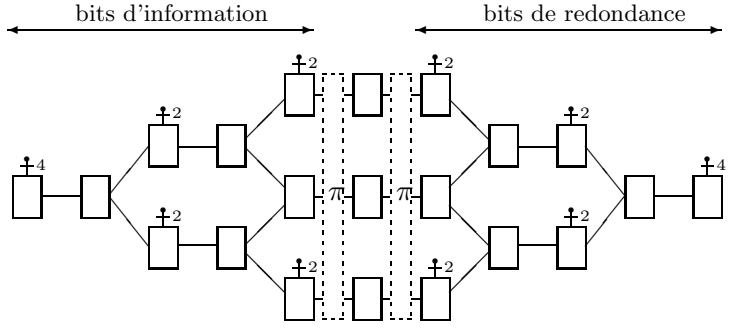


FIG. 3.13 – Code [24,12,8] à information répartie

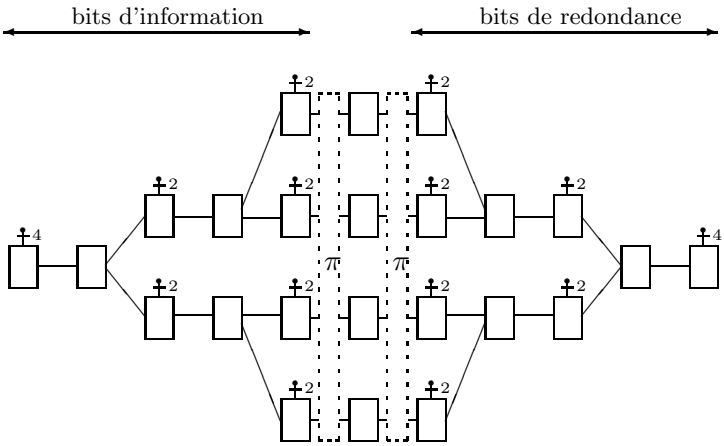


FIG. 3.14 – Code [32,16,8] à information répartie

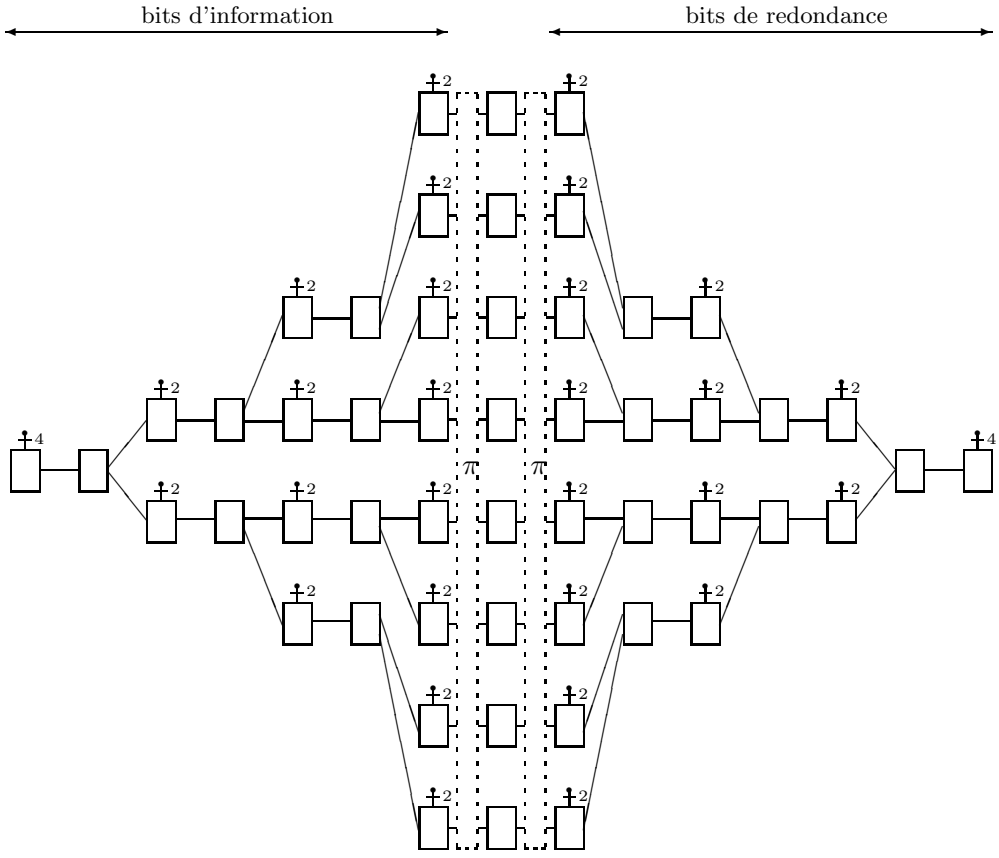


FIG. 3.15 – Code [64,32,12] à information répartie

Chapitre 4

Turbo codes parallèle

Le principe de construction des turbo codes présentés dans cette section a été initialement présenté par Cadic et al. dans [17]. L'approche diffère des turbo codes parallèle classiques dans le sens où le critère de choix des treillis composants devient prédominant et ne repose plus sur la complexité du treillis ou sur sa distance minimale. Le problème du choix du treillis composant dépend ici d'un nouveau critère qui est le polynôme de transition des poids du code et permet ainsi la réalisation de bons turbo codes utilisant des treillis comportant seulement 4 états.

4.1 Rappels sur les turbo codes

Cette nouvelle famille de code est l'œuvre de Berrou et al. [13]. Initialement, la construction était composée de deux codes convolutifs systématiques récursifs concaténés en parallèle et reliés *via* un entrelaceur. Ces codes permettent d'approcher de façon significative le limite de capacité de Shannon pour des taux d'erreur binaire de 10^{-5} notamment grâce à l'utilisation d'algorithmes de décodage itératif. À l'aide d'une variante de l'algorithme $BCJR$, on extrait de l'information sur chacun des codes et on l'échange entre eux. L'information ainsi extraite est appelée "extrinsèque" et est réinjectée à l'itération suivante dans un autre code composant. Ce processus pouvant se répéter plusieurs fois, on parle de décodage itératif. Ces algorithmes de décodage s'appliquent aux deux types de construction turbo : les turbo codes parallèle et les turbo codes série.

4.1.1 Turbo codes parallèle

Pour illustrer la construction parallèle, nous considérons ici l'utilisation de deux codes convolutifs récursifs systématiques (RSC1 et RSC2 sur la figure 4.1) que l'on représente à l'aide de treillis cycliques. Dans la pratique, on utilise souvent deux codes RCS1 et RSC2 identiques que l'on relie par un entrelaceur (Π sur la figure 4.1). Les bits d'informations X sont codés dans un premier temps par le codeur RSC1 pour fournir une première redondance Y_1 puis ces bits X sont entrelacés avant d'être encodés par le codeur RSC2 et délivrer une seconde redondance Y_2. Le message à transmettre est alors la concaténation de l'information X avec les redondances Y_1 et Y_2.

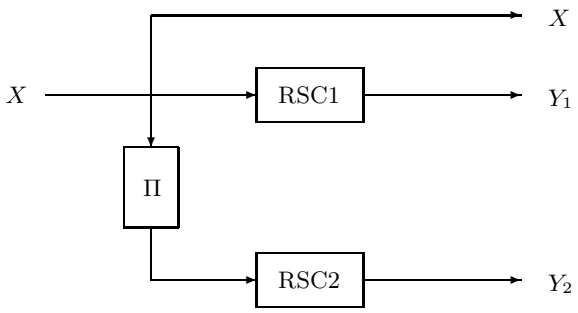


FIG. 4.1 – Turbo code parallèle

Le rendement global $\mathcal{R}_c$ du code ainsi construit est directement lié aux rendements R_1 et R_2 des deux codes RSC1 et RSC2 par la relation :

$$\mathcal{R}_c = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2}$$

Si R_1 et R_2 sont identiques, la relation devient :

$$\mathcal{R}_c = \frac{R_1}{2 - R_1}$$

4.1.2 Turbo codes série

Nous considérons là encore l'utilisation de deux codes convolutifs récursifs systématiques (RSC1 et RSC2 sur la figure 4.2 de rendements respectifs R_1 et R_2) que l'on représente à l'aide de treillis cycliques et que l'on relie *via* un entrelaceur (Π). Le premier code rencontré (RSC1) est appelé code externe alors que le second code (RSC2) est dit code interne. Les K bits d'informations X sont codés par le codeur RSC1, entrelacés à travers Π et ensuite recodés par le code RSC2. Le code externe

génère $N = \frac{K}{R_1}$ bits codés Y_1 et le code interne génère $J = \frac{N}{R_2}$ bits codés.

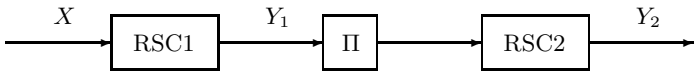


FIG. 4.2 – Turbo code série

Le rendement global $\mathcal{R}_c$ du code ainsi construit est directement lié aux rendements R_1 et R_2 des deux codes RSC1 et RSC2 par la relation :

$$\mathcal{R}_c = \frac{K}{J} = R_1 \times R_2$$

4.2 Critères de transition des poids sur un treillis

4.2.1 Motivation

Le but de cette partie est de déterminer un critère de choix pour les treillis cycliques qui serviront à l'élaboration de nos turbo codes. Ce critère doit respecter deux conditions bien spécifiques, à savoir qu'il doit permettre la construction de turbo codes ayant de bonnes distances minimales mais qu'il doit nous permettre de contrôler en partie la complexité de la structure. Une approche "classique" du choix des codes de base est par exemple d'augmenter la complexité des treillis de base en s'autorisant un plus grand nombre d'états dans chacun d'eux et donc d'obtenir des codes de base ayant des distances minimales plus élevées. Une autre solution permettant d'améliorer sensiblement le décodage, plus originale et moins onéreuse en terme de complexité, est celle proposée par Berrou et al. [11] avec l'adoption des turbo code duo-binaires. Ces derniers possèdent le même nombre d'états que les turbo codes classiques mais chacune des branches des treillis élémentaires comportent plus de bits que précédemment. Ce surplus de bits pour chacune des branches permet d'améliorer le décodage entre autre grâce au calcul des métriques de branche utilisées par les algorithmes de décodage qui prend ainsi en compte une plus grande quantité d'information provenant du canal. Les décisions réalisées à partir de ces métriques sont donc de meilleures qualité.
L'approche abordée ici est différente des précédentes car elle repose sur l'utilisation des polynômes de transition des poids. Un tel polynôme est bi-variable et se note :

$$W(X,Y) = \sum_{i,j} w_{i,j} X^i Y^j$$

Ce polynôme associe les poids i (poids de Hamming) des bits d'information avec les poids j des bits de redondance.
Pour illustrer l'intérêt de cette approche, considérons le cas d'un turbo code parallèle composé par la concaténation de deux codes C_1 et C_2. Si pour un vecteur d'information de poids de Hamming petit le code C_1 délivre une redondance avec un poids p_1 faible, on souhaite que le code C_2 délivre une redondance avec un poids p_2 elevé afin de compenser les autres poids et d'avoir globalement un poids important. Pour un code donné, connaissant son treillis, on est capable de déterminer le polnôme de transition des poids de ce code qui permet de connaître les relations entre les poids d'entrée et de sortie. Par conséquent, on recherche des codes pour lesquels la probabilité qu'un poids d'information faible soit associé avec un poids de redondance fort augmente avec la longueur du treillis.
En pratique, on constate que cette condition n'est pas directement liée à la distance minimale du code choisi, ce qui implique que l'on peut se limiter dans le nombre d'états des treillis constituants nos turbo codes.

4.2.2 Principe

On considère ici deux treillis comportant 16 états associés à des codes de rendement $\frac{1}{2}$, l'un de type duo-binaire qui permet de réaliser une distance minimale de 7 et l'un de type "bracelets" [20](Cf. section 3.2.1 et figure 4.3) autorisant une distance minimale de 8 (8 étant la meilleure distance minimale que l'on peut représenter sur 16 états pour un code de rendement $\frac{1}{2}$). À travers le comportement de ces treillis, nous allons mettre en avant deux aspects fondamentaux pour la recherche de bons codes composants pour les turbo-codes.

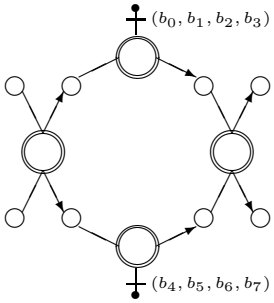


FIG. 4.3 – Un nœud du treillis bracelet

- ◇ Le premier aspect concerne la relation entre la distance minimale d'un code (et donc en partie le nombre d'états de son treillis minimal) et sa

capacité à associer un poids faible en information avec un poids fort en redondance. Pour cela, on note WT_{DB} le polynôme de transition des poids du treillis duo-binaire et WT_{Br} celui du treillis bracelet (ici, pour chaque nœud, $(b_0, \ldots, b_3)$ sont des bits d'information et $(b_4, \ldots, b_7)$ des bits de redondance). Un monôme de la forme X^iY^j signifie que pour un vecteur d'information de poids i, un vecteur de redondance de poids j est généré. Avec ces notations, considérons les premiers termes de chacun de ces deux polynômes pour des codes de paramètres [128,64,7] et [128,64,8]. Nous avons :

$$\begin{aligned} WT_{DB} &= 1 + X(16Y^{11} + 16Y^{13}) + X^2(16Y^5 + \ldots + 16Y^{25}) \\ &\quad + X^3(16Y^4 + \ldots + 16Y^{27}) \\ &\quad + X^4(16Y^4 + \ldots + 32Y^{27} + 4Y^{28}) + \ldots \end{aligned}$$

$$\begin{aligned} WT_{Br} &= 1 + X(32Y^7) + X^2(48Y^6 + \ldots + 208Y^{14}) \\ &\quad + X^3(64Y^5 + \ldots + 96Y^{24}) \\ &\quad + X^4(104Y^4 + \ldots + 4Y^{28}) + \ldots \end{aligned}$$

Un treillis est bien adapté à la construction turbo si, pour un poids d'information donné, la probabilité d'obtenir un poids de redondance élevé augmente. Par exemple, pour le treillis duo-binaire, pour un poids 1 en entrée, on peut avoir un poids 11 ou 13 en sortie. De plus si l'on augmente la longueur du treillis, il existe alors des poids de sortie supérieur à 13. En revanche pour le treillis bracelet, quelque soit la longueur du treillis, un poids 1 en information fournit un poids au plus 7 en redondance. Par conséquent, la concaténation de deux treillis bracelets conduit à un code de distance minimale au plus 15.
Il apparaît donc qu'un bon treillis de base pour la construction de turbo-codes ne dépend pas fondamentalement de la capacité correctrice du code composant utilisé. En effet, les treillis bracelets conduisent sur notre exemple à des codes composants de meilleure distance minimale que le treillis duo-binaire, cependant en raison de son incapacité à augmenter la fréquence des transitions "poids faible / poids fort", le treillis bracelet se révèle inadapté pour les constructions de type turbo.

◇ Le second aspect concerne l'influence du positionnement des bits sur une structure graphique. On considère ici le treillis bracelet précédent de polynôme énumérateur des poids WT_{Br} ainsi que le code utilisant le même treillis, mais pour lequel (b_0, b_1, b_4, b_5) sont des bits d'information et (b_2, b_3, b_6, b_7) des bits de redondance (ce code est de rendement $\frac{1}{2}$ uniquement pour un nombre pair de nœuds). Les premiers termes

de son polynôme énumérateur des poids WT_{Brp} sont donnés par

$$\begin{aligned} WT_{Br} &= 1 + X(39Y^{19}) + X^2(24Y^6 + \ldots + 32Y^{22}) \\ &\quad + X^3(96Y^5 + \ldots + 16Y^{25}) \\ &\quad + X^4(152Y^4 + \ldots + 4Y^{28}) + \ldots \end{aligned}$$

En réalisant ce simple repositionnement des bits sur le treillis, le code obtenu présente alors un comportement similaire aux treillis duo-binaire en terme de transitions de poids.

Le choix des positions pour les différents bits sur un treillis est donc très important, car nous venons de voir qu'un seul et même treillis peut conduire à la construction de deux codes distincts par simple repositionnement des bits, ces deux codes ayant des propriétes de transitions de poids totalement différentes.

Le choix d'un bon code composant pour les turbo-codes est donc un choix délicat car le critère de recherche ne concerne plus la distance minimale, mais une propriété de transition de poids. De plus, une approche trop simpliste par les treillis peut-être handicapante car un même treillis peut conduire à des capacités de diffusion des poids très différentes suivant le positionnement des bits.

La distance minimale n'étant pas ici l'aspect fondamental des treillis de base, nous avons donc opté pour des treillis à seulement 4 états dans un souci de minimisation de complexité. De plus, le positionnement des bits étant très important, cela signifie que le poinçonnage influe fortement sur ce phénomène de transition de poids. Aussi pour maîtriser au mieux son influence sur la distribution des poids du code globalement construit, nous poinçonnons dès le départ nos treillis en fonction du rendement global à atteindre et nous observons le comportement des treillis ainsi obtenus au travers des relations existantes entre les poids d'information et de redondance en fonction de la longueur du treillis.

4.3 Construction

Dans cette partie, nous mettons en application les différentes assertions formulées concernant les critères qualitatifs des codes composants pour les turbo-codes. Nous présentons dans un premier temps la méthode de construction de nos sections qui seront utilisées dans l'élaboration des treillis cycliques composants les turbo codes. Ces sections sont très importantes car elles interviennent pour la construction parallèle présentée dans ce chapitre mais aussi au chapitre suivant concernant la construction série.

4.3.1 Sections élémentaires des treillis

La section de base est construite à l'aide du code de Hamming étendu [8,4,4] de matrice génératrice

$$G_H = \begin{pmatrix} 1\,0\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0 \end{pmatrix}$$

De ce code nous pouvons déduire trois sections de treillis distinctes que l'on nomme H, H_g et H_d (Cf. figure 4.4, 4.5 et 4.6). Ces sections sont construites en considérant que les quatre bits d'information du Hamming composent l'étiquette et que les deux premiers bits de redondance sont associés à un état initial et les deux derniers à un état final. Les sections H_g et H_d découlent directement de la section H car elles correspondent respectivement au poinçonnage des deux bits de gauche et des deux bits de droite sur les étiquettes de la section H. Les matrices de transition associées à ces sections de treillis sont construites selon les mêmes conventions que celles obtenues dans la section 3.2.1, et sont données par :

$$T_H = \begin{pmatrix} 1 & Z^3 & Z^3 & Z^2 \\ Z^3 & Z^2 & Z^2 & Z \\ Z^3 & Z^2 & Z^2 & Z \\ Z^2 & Z & Z & Z^4 \end{pmatrix}$$

$$T_{H_g} = \begin{pmatrix} 1 & Z^2 & Z & Z \\ Z^2 & 1 & Z & Z \\ Z & Z & Z^2 & 1 \\ Z & Z & 1 & Z^2 \end{pmatrix}$$

$$T_{H_d} = \begin{pmatrix} 1 & Z & Z^2 & Z \\ Z & Z^2 & Z & 1 \\ Z^2 & Z & 1 & Z \\ Z & 1 & Z & Z^2 \end{pmatrix}$$

On rappelle que la notation Z^i signifie que la branche entre l'état indicé par le numéro de ligne de la matrice de transition et celui indicé par le numéro de colonne existe et que de plus son poids de Hamming est égal à i.
On rappelle également que les conventions de représentations utilisées pour les sections de treillis des figures 4.4, 4.5 et 4.6 sont identiques à celles utilisées pour la figure 3.5 de la partie 3.2.1.

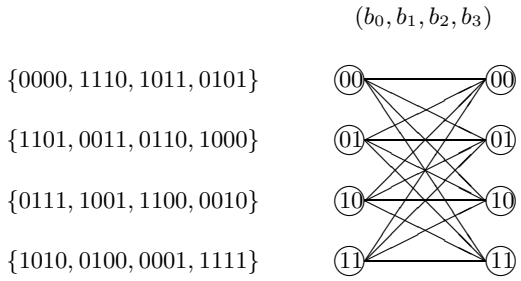


FIG. 4.4 – Section H

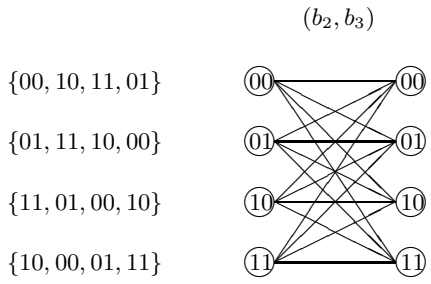


FIG. 4.5 – Section H_g

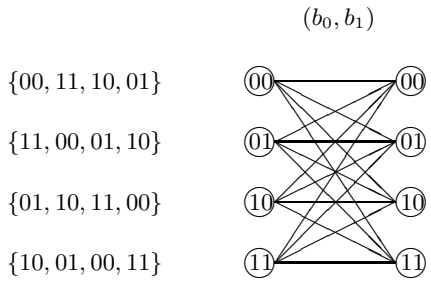


FIG. 4.6 – Section H_d

Les treillis construits par concaténation de ces différentes sections sont tous des treillis cycliques. La concaténation de deux sections distinctes S_a et S_b se note $(S_a|S_b)$ alors que la concaténation de deux sections identiques se note $(S_a|S_a)$ ou $(S_a)^2$.
La concaténation des treillis cycliques ainsi construits s'effectue alors par l'intermédiaire d'un code à répétition appliqué sur les bits d'information. On génère ainsi une "structure étoilée" dont le centre est un code à répétition,

les branches des permutations et les feuilles des codes à treillis cycliques comportant peu d'états (4 sur $\mathbb{F}_2$ et 16 sur $\mathbb{Z}_4$ si l'on utilise comme brique de base l'octacode de paramètres [8,4,6]). Les codes associés à ces treillis cycliques ont de plus la particularité de posséder un fort rendement ($\frac{3}{4}$ pour le Golay $[24, 12, 8]$) et une faible distance minimale (d_{min} vaut 2 sur $\mathbb{F}_2$ et 3 sur $\mathbb{Z}_4$). La figure 4.7 représente un exemple d'une telle structure (la figure 4.8 détaille une branche de cette étoile).

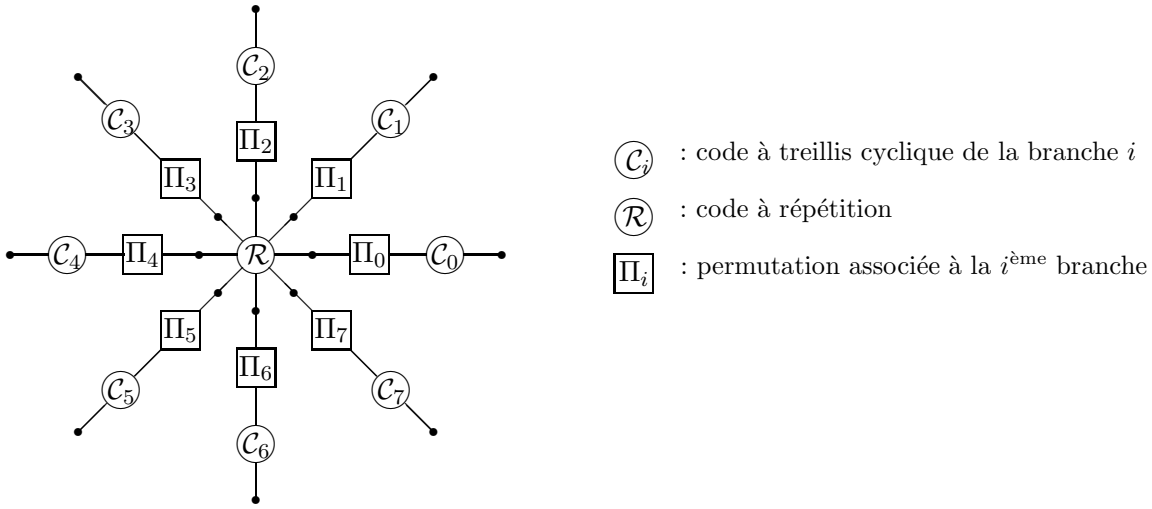


FIG. 4.7 – Graphe de Tanner d'un code étoile à 8 branches

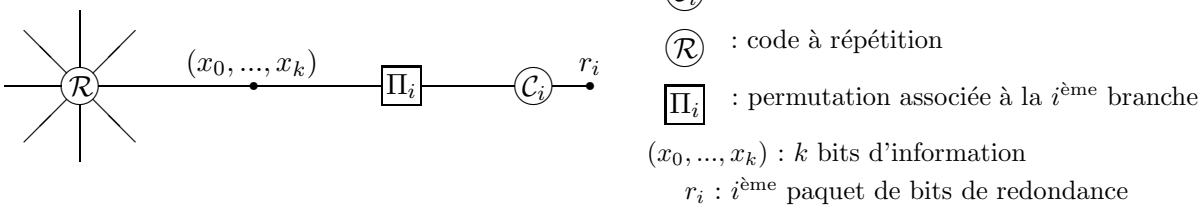


FIG. 4.8 – $i^{\text{ème}}$ branche de l'étoile

4.3.2 Exemples de construction

4.3.2.1 Code de Golay [24,12,8]

Pour illustrer l'encodage, nous utilisons la "boîte à outils" de sections de treillis présentée précédemment pour représenter le code de Golay

[24,12,8] sous une nouvelle forme de type turbo code, à savoir une triple concaténation parallèle de treillis cycliques à 4 états (Cf. figure 4.9). Chacun des treillis cycliques associé au code $\mathcal{C}$ transforme le vecteur de bits d'information $X = (x_0, \ldots, x_{k-1})$ (après une permutation différente pour chaque treillis) en un vecteur de bits de redondance. Le premier treillis (resp. le second et le troisième) fournissent des vecteurs de redondance $R_0 = \left(r_0^{(0)}, r_1^{(0)}, \ldots, r_{\frac{(k-1)}{3}}^{(0)}\right)$ (resp. R_1 et R_2). Le mot de code est alors la concaténation du vecteur d'information initial et des trois vecteurs de redondance obtenus.

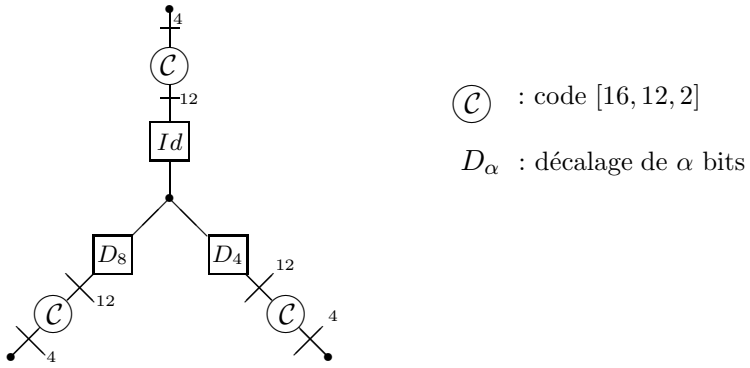


FIG. 4.9 – Code de Golay $[24, 12, 8]$ sous forme étoile

Chaque branche de l'étoile est en fait le treillis cyclique d'un code [16,12,2] construit par la concaténation $(H_g|H|H_g|H_d|H|H_d)$ qui correspond au treillis de la figure 4.10.
On peut également remarquer que dans [86] Sorger et al. présentent une forme en étoile du code de Golay [24,12,8]. Cependant la comparaison n'est pas immédiate entre ces deux représentations car celle de [86] correspond à un seul et unique treillis qui peut comporter jusqu'à 16 états par sections alors que notre représentation est une véritable représentation sous forme de turbo code *via* une concaténation parallèle multiple.

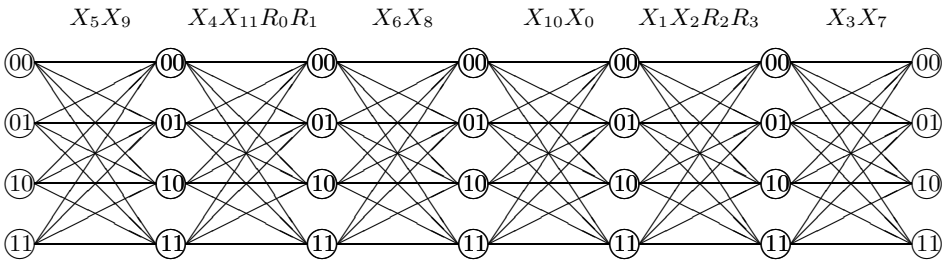


FIG. 4.10 – Treillis cyclique du code $[16, 12, 2]$

4.3.2.2 Code auto-dual [32,16,8]

Un code auto-dual de paramètres [32,16,8] peut se mettre sous la forme de la concaténation de quatre treillis identiques à quatre états. Chacun de ces treillis permet de générer quatre bits de redondance.

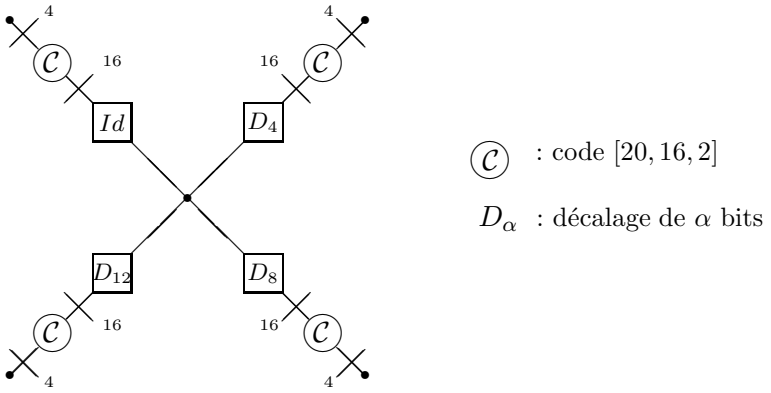


FIG. 4.11 – Code [32,16,8] sous forme étoile

Le code $\mathcal{C}$ est ici un code de paramètres $[20, 16, 2]$ dont le treillis cyclique peut se mettre sous la forme :

$$\mathcal{T}reillis(\mathcal{C}) = (H|H_g|H_g|H_g|H|H_g|H_g|H_g)$$

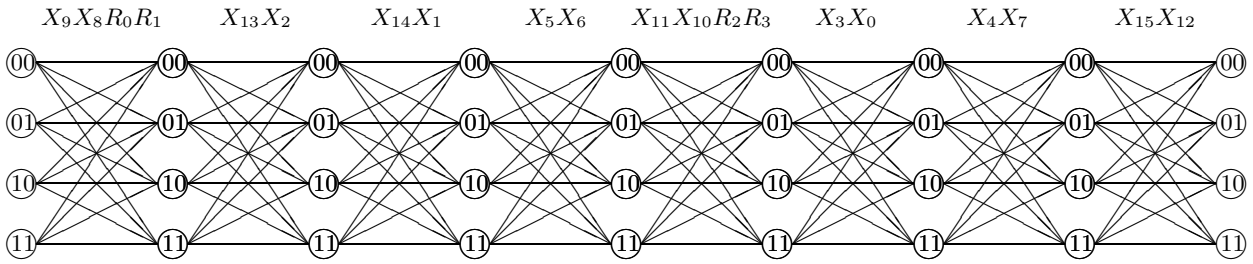


FIG. 4.12 – Treillis cyclique du code $[20, 16, 2]$

Le code de paramètres [20,16,2] ainsi obtenu a pour matrice génératrice $G_{[20,16,2]} = (Id_{12} \mid P)$ avec

$$ {}^tP = \begin{pmatrix} 1&0&0&1&1&0&0&1&1&0&1&1&1&1&1&1 \\ 0&1&1&1&1&1&1&1&0&1&1&0&0&1&1&0 \\ 1&0&1&1&1&1&1&1&1&0&0&1&1&0&0&1 \\ 0&1&1&0&0&1&1&0&0&1&1&1&1&1&1&1 \end{pmatrix} $$

et son polynôme énumérateur des poids est :

$$\begin{aligned} W(Z) &= 1 + 14Z^2 + 64Z^3 + 301Z^4 + 960Z^5 + 2472Z^6 + 4928Z^7 \\ &\quad + 7634Z^8 + 10432Z^9 + 11924Z^{10} + 10432Z^{11} + 7634Z^{12} \\ &\quad + 4928Z^{13} + 2472Z^{14} + 960Z^{15} + 301Z^{16} + 64Z^{17} + 14Z^{18} + Z^{20} \end{aligned}$$

4.3.2.3 Codes construits

Les deux tableaux 4.1 et 4.2 récapitulent quelques uns des codes construits. Il est à noter que toutes les permutations utilisées sont des permutations affines, donc de la forme :

$$i \mapsto (a * i + b) \mod k \text{ pour } (a, b) \in \mathbb{Z} \times \mathbb{Z}$$

exceptées les deux permutations suivantes :

$$\begin{aligned} \Pi_0 &= (5, 9, 4, 11, 6, 8, 10, 0, 1, 2, 3, 7) \\ \Pi_1 &= (9, 8, 13, 2, 14, 1, 5, 6, 11, 10, 3, 0, 4, 7, 15, 12) \end{aligned}$$

Les différentes distances minimales mentionnées sont obtenues à l'aide d'un logiciel de calcul qui fonctionne à partir des matrices génératrices sous forme systématique des codes considérés. Ce logiciel est l'œuvre de Canteaut et est dérivé d'un algorithme présenté par Canteaut et al. dans [23].

code de paramètres $[n, k, d]$	nombre de branches	couple (a, b) pour chacune des branches	composition du treillis de branche
[60,20,8]	4	Id ;(19,7) ;(23,0) ;(31,7)	$(H\|H_g)^5$
[120,40,16]	4	Id ;(19,7) ;(23,0) ;(31,7)	$(H\|H_g)^{10}$
[240,80,32]	4	Id ;(19,7) ;(23,7) ;(31,7)	$(H_g\|H)^{20}$

TAB. 4.1 – Codes de rendement $\frac{1}{3}$

code de paramètres $[n, k, d]$	nombre de branches	couple (a, b) pour chacune des branches	composition du treillis de branche
[24,12,8]	3	Π_0 ; $D_4 \circ \Pi_0$;$D_8 \circ \Pi_0$	$(H_g\|H\|H_g\|H_d\|H\|H_d)$
[32,16,8]	4	Π_1 ; $D_4 \circ \Pi_1$;$D_8 \circ \Pi_1$;$D_{12} \circ \Pi_1$	$(H\|H_g^3)^3$
[128,64,12]	4	Id ;(37,5) ;(53,7) ;(19,3)	$(H_g^2\|H\|H_g)^8$
[256,128,18]	4	Id ;(31,5) ;(53,5) ;(73,0)	$(H_g^2\|H\|H_g)^{16}$

(en caractères gras les codes auto-duaux extrémaux)

TAB. 4.2 – Codes de rendement $\frac{1}{2}$

4.4 Décodage

Les premiers essais de décodage ont dans un premier temps été réalisés en utilisant une simple concaténation parallèle pour laquelle les treillis cycliques composant le turbo code sont élaborés uniquement en utilisant la section H_γ (ce type de section intervient essentiellement dans le cadre des constructions séries et sa description est donnée dans la partie 5.1.1 du chapitre suivant). Les algorithmes utilisés sur chacun des treillis pour l'extraction des informations extrinsèques sont de type $SOVA$. Malheureusement le décodage des concaténations multiples reste une question ouverte et n'a pu être réalisé faute de temps.

4.4.1 Codes de rendement $\frac{1}{3}$

La figure 4.13 présente des courbes de décodage pour des codes de rendement $\frac{1}{3}$. Par convention, on note P3 le code de paramètres [300,100] et P6 le code de paramètre [600,200]. Le résultat immédiat qui ressort de ces courbes est que ces codes nécessitent un nombre relativement important d'itérations avant de commencer à réaliser "l'effet turbo". L'une des questions posées concernant ce phénomène est de comprendre si le fait d'utiliser

des treillis à seulement 4 états (et donc possédant de petites distances minimales lorsqu'ils sont considérés indépendamment les uns des autres) n'est pas un facteur retardant pour le déclenchement de l'effet turbo.

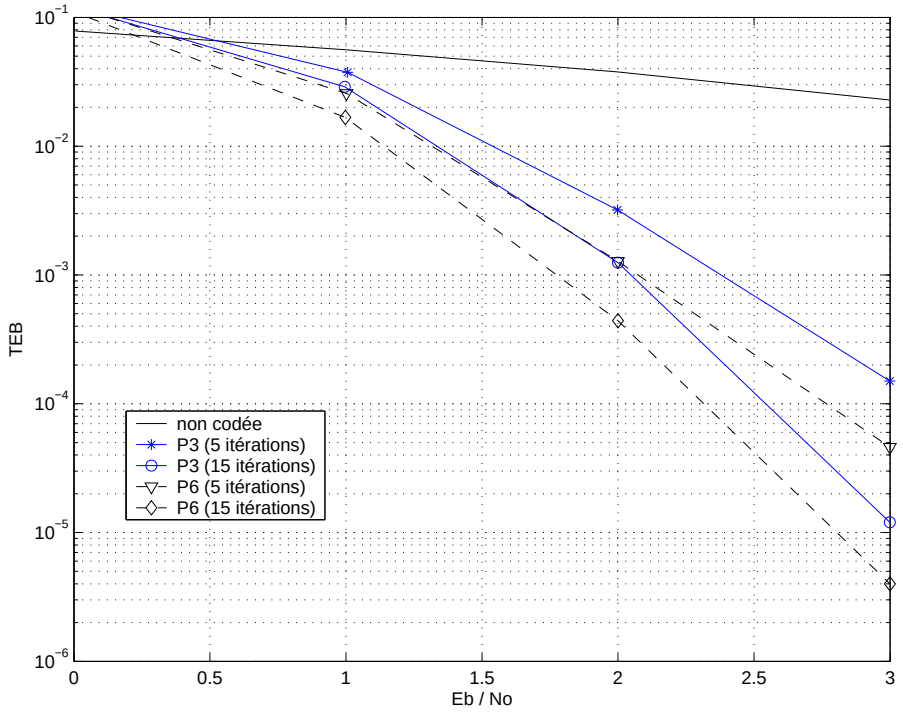


FIG. 4.13 – Décodage des codes P3 [300,100] et P6 [600,200]

4.4.2 Codes de rendement $\frac{1}{2}$

La figure 4.14 correspond à la comparaison de codes [400,200] construits selon trois méthodes distinctes. On note P4 le code de paramètre [400,200] construit avec nos sections de treillis. La courbe de TEB présentée pour ce code l'est pour un canal $AWGN$ et nécessite 15 itérations. Les courbes des codes CT (3,1), qui est la concaténation de 3 arbres, et TC (21,37), qui est un turbo code à 16 états, sont issues de [79]. Elles nécessitent chacune 18 itérations et correspondent aussi à un canal $AWGN$.

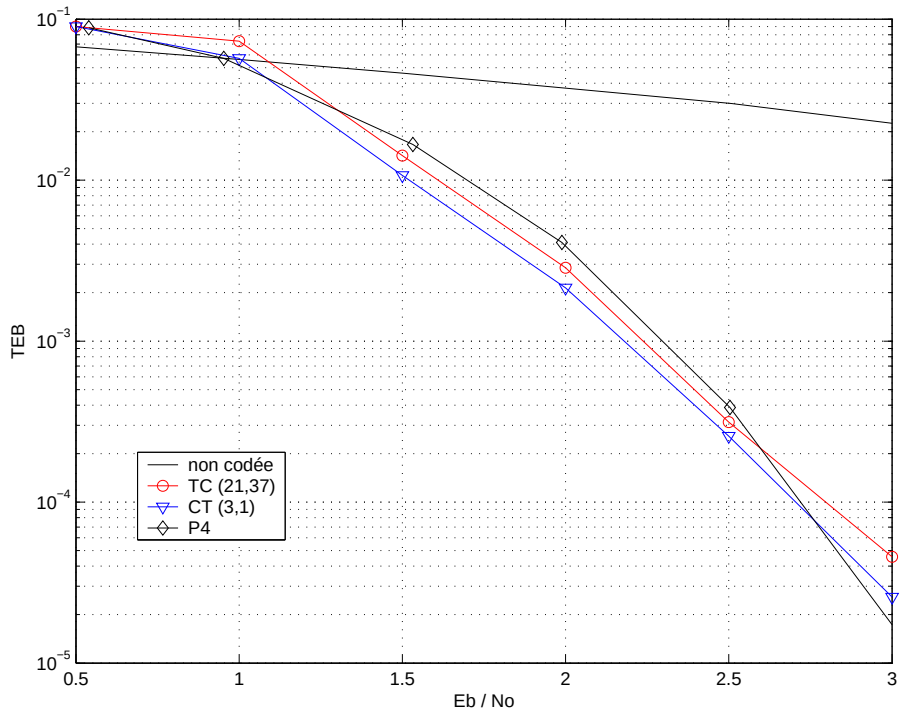


FIG. 4.14 – Comparaison de décodage de codes [400,200]

4.5 Représentation des codes cortex en étoile

L'un des enseignements de l'étude des codes "cortex" de Carlach et al. [25][26] est la difficulté de les décoder à partir de leur représentation classique. Cependant certains de ces codes, notamment ceux de distance minimale 8 sur $\mathbb{F}_2$ ou 12 sur $\mathbb{Z}_4$ peuvent se représenter sous la forme de "treillis bracelets"[20] comportant respectivement 16 et 256 états que l'on sait décoder. L'inconvénient de ces "treillis bracelets" est qu'ils ne peuvent pas être associés à tout code "cortex" en raison de leur dépendance avec les permutations utilisées. Malgré ces difficultés, cette famille de codes demeure très intéressante car les codes "cortex" atteignent la borne de Gilbert-Varshamov comme l'ont montré Olocco et Tillich [74]. Nous présentons donc une nouvelle description des codes "cortex" utilisant la quasi-cyclicité de certains de ces codes.

4.5.1 Quelques notions sur les codes circulants

Soit $\mathcal{C}$ un code $[n, k, d]$ de matrice génératrice $(I|R)$ où I est la matrice identité de dimension k et R une matrice circulante de dimension k.

Définition 4.5.1 *Une matrice circulante $k \times k$ est une matrice de la forme*

$$R = \begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{k-1} \\ r_{k-1} & r_0 & r_1 & \cdots & r_{k-2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ r_1 & r_2 & r_3 & \cdots & r_0 \end{pmatrix}$$

Définition 4.5.2 *Un code $\mathcal{C}$ possédant une matrice génératrice de la forme $(I|R)$, où R est une matrice circulante, est appelé code pur doublement circulant.*

On introduit maintenant la notion de codes circulants par bloc. Soit $(M_i)_{i\in\mathbb{N}}$ un ensemble de matrices carrées, non nécessairement identiques, de dimension m, où m est un diviseur de k ($k = l \times m$).

Définition 4.5.3 *On dit qu'un code $\mathcal{C}$, de matrice génératrice $(I|R)$ est un code circulant par bloc si R est une matrice bloc de la forme*

$$\begin{pmatrix} M_0 & M_{l-1} & M_{l-2} & \cdots & M_1 \\ M_1 & M_0 & M_{l-1} & \cdots & M_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ M_{l-1} & M_{l-2} & M_{l-3} & \cdots & M_0 \end{pmatrix}$$

R est appelée Matrice Circulante par Bloc ou MCB.

Définition 4.5.4 *On dit qu'un code $\mathcal{C}$, de matrice génératrice $(I|R)$ est un code équivalent par permutation affine à un code circulant par bloc si R est une matrice bloc telle qu'il existe une permutation affine P vérifiant "$R = P \times T$" où T est une matrice circulante par bloc.*

4.5.2 Aspect circulant des codes "cortex"

Le but de cette partie est de montrer le lien existant entre le code de base utilisé pour les codes cortex et l'aspect circulant des codes ainsi obtenus lorsque les permutations de la construction sont affines. Ce phénomène peut se résumer sous la forme de la remarque suivante :

Remarque 4.5.1 *Les matrices génératrices des codes obtenus par la construction cortex utilisant un code de base $\mathcal{C}_b$ de paramètres $[n_b, k_b, d_b]$ et des permutations affines sont des MCB dont les blocs sont de dimension multiple de k_b.*

En préambule de la démonstration de l'aspect circulant des codes "cortex" construits avec des permutations affines, on énonce la proposition suivante qui intervient dans la démonstration :

Proposition 4.5.1 *Soient U une matrice équivalente par permutation affine à une matrice V circulante par bloc (chaque bloc étant de dimension k_b) et $\mathcal{R}$ une matrice diagonale de la forme*

$$\begin{pmatrix} R_0 & 0 & 0 & \cdots & 0 \\ 0 & R_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & R_{t-1} \end{pmatrix}$$

où les R_i sont des matrices carrées identiques de dimension k_b.
Alors, le produit $U\mathcal{R}$ est une matrice équivalente par permutation affine à une MCB.

Démonstration : Par définition, il existe une matrice de permutation P (où P est affine) telle que $U = PV$ où V est une *MCB*. Nous avons donc :

$$U\mathcal{R} = (PV)\mathcal{R} = P(V\mathcal{R})$$

où V est de la forme

$$\begin{pmatrix} \alpha_0 & \alpha_{t-1} & \alpha_{t-2} & \cdots & \alpha_1 \\ \alpha_1 & \alpha_0 & \alpha_{t-1} & \cdots & \alpha_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \alpha_{t-1} & \alpha_{t-2} & \alpha_{t-3} & \cdots & \alpha_0 \end{pmatrix}$$

et les α_i sont des matrices carrées de dimension k_b. Par conséquent,

$$V\mathcal{R} = \begin{pmatrix} \alpha_0 R & \alpha_{t-1} R & \alpha_{t-2} R & \cdots & \alpha_1 R \\ \alpha_1 R & \alpha_0 R & \alpha_{t-1} R & \cdots & \alpha_2 R \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \alpha_{t-1} R & \alpha_{t-2} R & \alpha_{t-3} R & \cdots & \alpha_0 R \end{pmatrix} = \begin{pmatrix} \beta_0 & \beta_{t-1} & \beta_{t-2} & \cdots & \beta_1 \\ \beta_1 & \beta_0 & \beta_{t-1} & \cdots & \beta_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \beta_{t-1} & \beta_{t-2} & \beta_{t-3} & \cdots & \beta_0 \end{pmatrix}$$

avec $\beta_i = \alpha_i R$ pour $i \in \{0, ..., t-1\}$.
$V\mathcal{R}$ est donc une *MCB*.

Il s'en suit que $U\mathcal{R} = PV\mathcal{R}$ est équivalente par permutation affine à une *MCB*. ■

Pour un encodeur cortex composé de s étages comportant chacun t boîtes d'un code de base $\mathcal{C}_b$ de paramètres $[n_b, k_b, d_b]$ et de matrice génératrice $(Id_{k_b} \mid R)$, on note $\mathcal{P}(\pi_i)$ la matrice de la permutation reliant les étages i et $i+1$. De plus, on note

$$\mathcal{R} = \begin{pmatrix} R_0 & 0 & 0 & \cdots & 0 \\ 0 & R_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & R_{t-1} \end{pmatrix}$$

la matrice représentant la transition d'étage, où les R_j ($j \in \{0, ..., t-1\}$) sont tous identiques et égaux à la partie redondante R de la matrice génératrice du code de base $\mathcal{C}_b$. $\mathcal{R}$ est donc une *MCB*.

Proposition 4.5.2 *Soit $\mathcal{C}$ un code cortex composé de s étages comportant chacun t boîtes d'un code de base $\mathcal{C}_b$ de paramètres $[n_b, k_b, d_b]$ reliés par des permutations affines. La matrice génératrice d'un tel code cortex est équivalente par permutation affine à une MCB.*

Démonstration : On prend comme hypothèse de récurrence que la matrice

$$M_i = M_{i-1} \times \begin{pmatrix} R_0 & 0 & 0 & \cdots & 0 \\ 0 & R_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & R_{t-1} \end{pmatrix} \times \mathcal{P}(\pi_{i-1}) = M_{i-1} \times \mathcal{R}_{i-1}$$

(où $\mathcal{P}(\pi_{i-1})$ est une permutation affine) obtenue après i ($i \prec s$) étages de cortex est équivalente par permutation affine à une *MCB*.

- ◇ On vérifie l'hypothèse pour le premier étage.
  On obtient une matrice de la forme

$$M_1 = M_0 \times \begin{pmatrix} R_0 & 0 & 0 & \cdots & 0 \\ 0 & R_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & R_{t-1} \end{pmatrix} \times \mathcal{P}(\pi_0) = M_0 \times \mathcal{R}_0$$

  qui est une matrice équivalente par permutation affine à une *MCB* car les R_j ($j \in \{0, ..., t-1\}$) sont tous identiques.

◇ Supposons l'hypothèse vérifiée jusqu'à l'étage i.
Le passage de l'étage i vers l'étage $i+1$ s'effectue par la relation $M_{i+1} = M_i\mathcal{R}_i$. Or, d'après la proposition 3, $M_i\mathcal{R}$ est équivalente par permutation affine à une *MCB*. Donc, M_{i+1} est équivalente par permutation affine à une *MCB*. ■

4.5.3 Cas des codes "cortex" utilisant le Hamming [8,4,4]

On considère ici les codes "cortex" construits selon le schéma de la figure 1, utilisant des codes de Hamming étendu [8,4,4] comme code de base et des permutations de la forme :

$$i \mapsto (a * i + b) \mod k \text{ pour } (a,b) \in \mathbb{Z} \times \mathbb{Z}.$$

D'après la partie précédente (et notamment la proposition 4), ces codes ont en commun d'être des codes circulants par bloc. De plus, ces codes sont circulants par bloc de dimension au moins 4 (qui est la dimension du code de base choisi). Autrement dit, si $\mathcal{C}$ est un code "cortex" $[n,k,d]$ de matrice génératrice $(I|R)$, on peut écrire, pour $k = 4 \times l$:

$$R = \begin{pmatrix} M_0 & M_{l-1} & M_{l-2} & \cdots & M_1 \\ M_1 & M_0 & M_{l-1} & \cdots & M_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ M_{l-1} & M_{l-2} & M_{l-3} & \cdots & M_0 \end{pmatrix}$$

où $(M_i)_{i\in\{0,1,\ldots,l-1\}}$ sont des matrices carrées de dimension 4. Cette spécificité de la matrice génératrice implique que la connaissance des 4 premières colonnes de la partie redondante R suffit pour déterminer entièrement la matrice et donc le code associé. En d'autres termes, le procédé d'obtention de la matrice complète à partir d'un bloc quelconque de 4 colonnes consiste à concaténer ce bloc avec ce même bloc ayant subi un décalage de 4 sur ses lignes. On recommence le procédé avec le nouveau bloc de 4 colonnes ainsi obtenu jusqu'à atteindre la dimension du code.
D'un point de vue plus graphique, il suffit donc de générer le treillis cyclique du code de paramètres $[k+4, k, d_{min} \preceq 2]$ et de matrice génératrice $(I|P)$ où ${}^tP = (M_0M_1...M_{l-1})$. Si l'on note $\mathcal{T}$ le treillis associé à un tel code, alors nous pouvons écrire

$$\max_{i\in\{0,1,\ldots,k+4\}} c_i(\mathcal{T}) \preceq \frac{1}{2} \times \min(k,4) = 2.$$

Par conséquent le nombre d'états maximal pour une section de treillis est borné supérieurement par 4. Les codes "cortex" utilisant un code de Hamming comme code de base peuvent donc se mettre sous la forme de la concaténation de treillis cycliques identiques à 4 états reliés par des permutations qui sont en fait de simples décalages.

4.5.4 Représentation du code étoile

Soit D_α l'application définie comme suit :

$$\begin{array}{rccc} & \{0,1,2,...,(dimension-1)\} & \longrightarrow & \{0,1,2,...,(dimension-1)\} \\ D_\alpha : & i & \longmapsto & (i+\alpha) \mod dimension \end{array}$$

Le principe de concaténation présenté dans la partie précédente pour les codes "cortex" consiste en fait à construire le code global comme étant la concaténation de codes $\mathcal{C}_b$ de paramètres $[k+4, k, d \preceq 2]$, le passage d'un code $\mathcal{C}_b$ à son voisin immédiat étant réalisé par la permutation D_4.

Proposition 4.5.3 *Soit $\mathcal{C}_g[n_g, k_g, d_g]$ le code de base binaire utilisé dans la construction cortex d'un code $\mathcal{C}[n, k, d]$. Alors, si $k \succeq 2^{k_g}$, il existe un code de branche pour la représentation étoile ayant comme paramètres $[k+m, k, d_{min} \preceq 2]$ où m est un multiple de k_g.*

Démonstration : D'après la remarque 4.5.1, les codes cortex construits sont circulants par blocs de dimensions multiples de k_g.
On peut donc toujours représenter le code sous forme d'une étoile où chaque branche est de dimension k et fournit $u \times k_g (u \in \mathbb{N})$ bits de redondance.
Or, le nombre de vecteurs binaires que l'on peut construire avec $u \times k_g$ bits est 2^{uk_g}. Donc, si $k \succeq 2^{uk_g}$, cela implique nécessairement qu'il existe au moins deux vecteurs de la matrice génératrice du code possédant une même partie redondante. La somme de ces deux vecteurs donne une partie redondante nulle et une partie information de poids 2 (car la matrice est systématique).
Le code de branche construit est donc de distance minimale au plus 2. ■

Dans cette partie, on présente les turbo-codes étoiles associés aux codes "cortex" à travers l'exemple du code de paramètres [64,32,12] qui se construit en 3 couches et utilise la permutation "$i \mapsto 19i \mod 32$".
On définit le code $\mathcal{C}$ de paramètres [36,32,1] par sa matrice génératrice

$(Id_{32} \mid P)$ avec

$$ {}^tP = \begin{pmatrix} 0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,1\,0 \\ 1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,1\,1\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\,1 \\ 1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,0\,1 \end{pmatrix} $$

D'après la partie précédente ce code peut se représenter sous la forme d'un treillis cyclique comportant au plus 4 états. La figure 4.15 présente la forme étoile du code [64,32,12] décomposé en 4 "hyper-branches" reliées entre elles par des décalages multiples de 8. Chacune de ces "hyper-branches" se ramifie en deux branches composées d'un treillis cyclique du code $\mathcal{C}$[36,32,1]. Les "hyper-branches" sont importantes dans la représentation car le code qu'elles représentent est un code de paramètres $[40, 32, 2]$ qui assure que sur chacune des hyper-branches, chaque bit d'information apparaît dans l'équation de parité d'au moins un bit de redondance fourni par cette hyper-branche. Cela est un atout pour le décodage de type turbo car cela assure que tous les bits d'information verront leurs métriques évoluer *via* l'influence des bits de redondance de chaque hyper-branche.
Ces hyper-branches présentent un second intérêt en terme de complexité. En effet, la complexité d'un treillis est en grande partie due à la distance minimale du code à représenter, mais aussi de sa distribution des poids. Plus la distance minimale est importante et plus il est difficile de trouver un treillis cyclique comportant peu d'états. Or les hyper-branches, quelque soit le code global construit à partir des codes "cortex", sont associées avec des codes de distance minimale constante égale à 2. Par conséquent il serait intéressant d'étudier de plus près ces codes de distance minimale 2 et d'essayer de les représenter sous forme de treillis comportant peu d'états. Cela permettrait d'avoir une représentation générique des codes cortex qui soit adaptée aux techniques de décodage itératif.

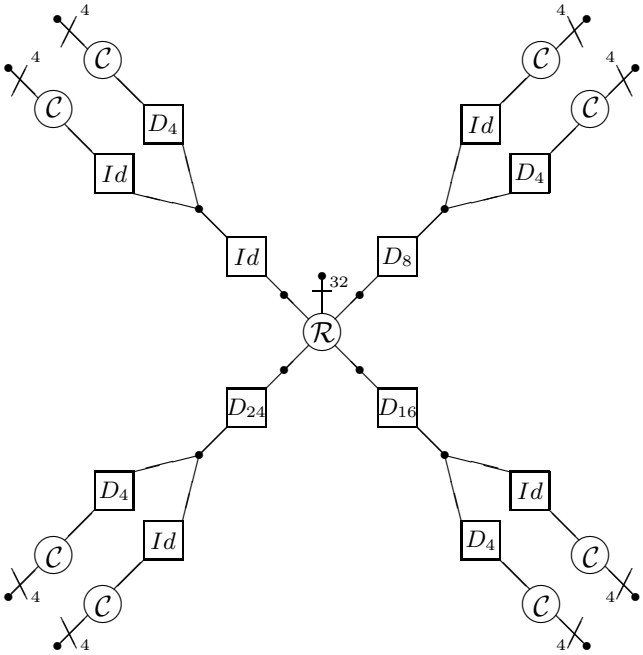


FIG. 4.15 – Turbo-code étoile associé au code cortex [64,32,12]

Chapitre 5

Turbo codes série

Le choix de cette construction découle de la recherche de codes de bonne distance minimale à partir des codes "cortex" [25]. Pour obtenir d'importantes distances minimales, cette construction nécessite l'utilisation d'un grand nombre d'étages, ce qui malheureusement rend le décodage très complexe. L'augmentation du nombre d'étages permet une meilleure diffusion des bits horizontalement mais influence trop peu la diffusion verticale des bits. Pour remédier en partie à ce problème, on se propose de rajouter des connexions verticales dans la construction cortex. Cela revient finalement à une concaténation série de treillis cycliques identiques dont les sections sont construites à l'aide du code de Hamming étendu [8,4,4]. Comme pour les codes cortex, seuls les bits positionnés sur les bords de la structure sont émis. Cependant du fait du "mélange vertical" dû à la structure treillis effectué sur les bits de chaque étage, il n'est plus nécessaire d'augmenter le nombre d'étages pour augmenter la distance minimale. Avec seulement 3 étages, les codes obtenus présentent de bonnes distances minimales. En raison des connexions verticales, les codes de Hamming deviennent des sections de treillis à 4 états que l'on peut utiliser de différentes façons liées au choix du positionnement des bits d'information et de redondance sur la section.

5.1 Construction

5.1.1 Principe

On considère 4 positionnements différents des bits sur la section H de la figure 4.4 qui conduisent à 4 sections distinctes. Chaque section porte 4 bits, 2 d'information (x_0 et x_1) et 2 de redondance (y_0 et y_1). Les 4 sections utilisées sont alors H_α, H_β, H_γ et H_δ qui correspondent à la section

de la figure suivante pour laquelle les bits (b_0, b_1, b_2, b_3) sont respectivement étiquetés par (y_0, y_1, x_0, x_1), (x_0, x_1, y_0, y_1), (x_0, y_0, y_1, x_1) et (y_0, x_0, x_1, y_1). Muni de cette "boîte à outils" de sections de treillis nous construisons des treillis cycliques à 4 états comme nous l'avons fait au chapitre précédent. Cependant le critère de choix de ces treillis ne repose plus uniquement sur le polynôme de transition des poids. Ce critère est bien evidemment toujours très important mais le critère principal provient d'une propriété très particulière du code de Hamming étendu [8,4,4]. On considère sa matrice génératrice sous la forme

$$G_{[8,4,4]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

On note $c = (c_0, c_1, \ldots, c_7)$ un mot de code quelconque. Si l'on forme les quatres couples de bits (c_0, c_1), (c_2, c_3), (c_4, c_5) et (c_6, c_7) alors, excepté pour le mot nul, nous avons toujours au moins 3 de ces couples qui sont différents de (0,0). Ce phénomène est prépondérant pour la construction des turbo codes série si l'on considère la figure 5.1 qui schématise le fonctionnement d'une section de l'un des treillis cycliques composant le turbo code série.

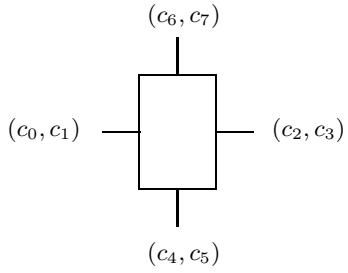


FIG. 5.1 – Schématisation d'une section de treillis

Lors de la construction série, le couple (c_0, c_1) correspond à des bits d'entrée, le couple (c_2, c_3) à des bits de sortie qui deviendront des bits d'entrée pour le treillis suivant, alors que les couples (c_4, c_5) et (c_6, c_7) correspondent aux bits d'état du treillis. Comme au moins 3 couples sur les quatre sont non nuls, cela implique que l'on ne peut pas stopper la propagation des bits non-nuls et donc borner rapidement la distance minimale. Par exemple si le couple (c_2, c_3) est nul, cela signifie que cette section de treillis ne propage pas de 1 au treillis suivant, mais les autres couples sont

nécessairement non nuls et donc les 1 se propagent sur le treillis même vers une autre section qui elle pourra être dans une configuration où le couple (c_2, c_3) sera non nul. Par conséquent des bits non nuls pourront finalement être transmis au treillis suivant de la concaténation série et augmenter ainsi l'espérance d'obtenir des bits non nuls sur le dernier étage de la structure série.
Nous n'avons pas obtenu d'autres codes binaires possédant cette propriété, en revanche sur $\mathbb{Z}_4$ l'octacode de matrice génératrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 3 & 3 & 2 & 1 \\ 0 & 0 & 1 & 0 & 3 & 1 & 3 & 2 \\ 0 & 0 & 0 & 1 & 3 & 2 & 1 & 3 \end{pmatrix}$$

et le code sur $\mathbb{Z}_8$ de matrice génératrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 5 & 7 & 4 \\ 0 & 1 & 0 & 0 & 4 & 1 & 5 & 7 \\ 0 & 0 & 1 & 0 & 7 & 4 & 1 & 5 \\ 0 & 0 & 0 & 1 & 5 & 7 & 4 & 1 \end{pmatrix}$$

possèdent cette même propriété.
Le tableau 5.1 récapitule le nombre de mots de chacun des 3 codes cités pour lesquels cette propriété de 3 couples de bits sur 4 non nuls est vérifié. Plus le pourcentage de cas de figure où les 4 couples sont simultanément non

Anneau	0 couple non nul	3 couples non nuls	4 couples non nuls
$\mathbb{Z}_2$	1	12 (75 %)	1 (18.75 %)
$\mathbb{Z}_4$	1	60 (23.44 %)	195 (76.17 %)
$\mathbb{Z}_8$	1	252 (6.15 %)	3843 (93.82 %)

TAB. 5.1 – Répartition des couples de bits non nuls

nuls, plus le code est intéressant selon nos critères pour la construction de turbo codes série. Cependant cela implique que la section de treillis de base utilisée aura une complexité plus importante en terme d'états (16 sur $\mathbb{Z}_4$ et 256 sur $\mathbb{Z}_8$). Néanmoins le fait de travailler sur $\mathbb{Z}_4$ ou $\mathbb{Z}_8$ est intéressant dans l'optique de réaliser des modulations codées.

5.1.2 Représentation graphique

Les turbo codes série construits sont des codes correcteurs d'erreurs en bloc systématiques de longueur $n = 2k$ comportant k bits d'information.

Ces codes se présentent sous la forme d'une double concaténation série de petits codes de base $\mathcal{C}_b[n_b, k_b, d_b]$ (la figure 5.2 utilise uniquement le code de Hamming [8,4,4] comme code de base).
Les codes de base sont dans un premier temps concaténés en série suivant un axe vertical engendrant de ce fait un treillis cyclique à $2^{\frac{k_b}{2}}$ états dans le cas binaire. Dans un second temps, les treillis ainsi obtenus sont concaténés en série selon un axe horizontal et séparés les uns des autres par des permutations (Π_i sur les figures 5.2 et 5.3) non nécessairement identiques.
Le processus d'encodage systématique se déroule de la façon suivante : le premier étage transforme le vecteur de bits d'information à encoder $X = (x_0, x_1, ..., x_{k-1}) = (x_0^{(0)}, x_1^{(0)}, ..., x_{k-1}^{(0)}) = X^{(0)}$ en un vecteur de bits de redondance $R^{(0)} = (r_0^{(0)}, r_1^{(0)}, ..., r_{k-1}^{(0)})$. Les éléments de ce vecteur redondance subissent alors une première permutation Π_0 pour obtenir $R^{(1)} = (r_{\pi(0)}^{(0)}, r_{\pi(1)}^{(0)}, ..., r_{\pi(k-1)}^{(0)})$. On procède ainsi pour les 3 étages et les 2 permutations qui composent la structure. Le mot de code correspond alors à la concaténation du vecteur X de bits d'information avec le vecteur $R^{(2)}$ de bits de redondance. On se limite volontairement à l'utilisation de 3 étages et 2 permutations dans le but d'améliorer la qualité de décodage.

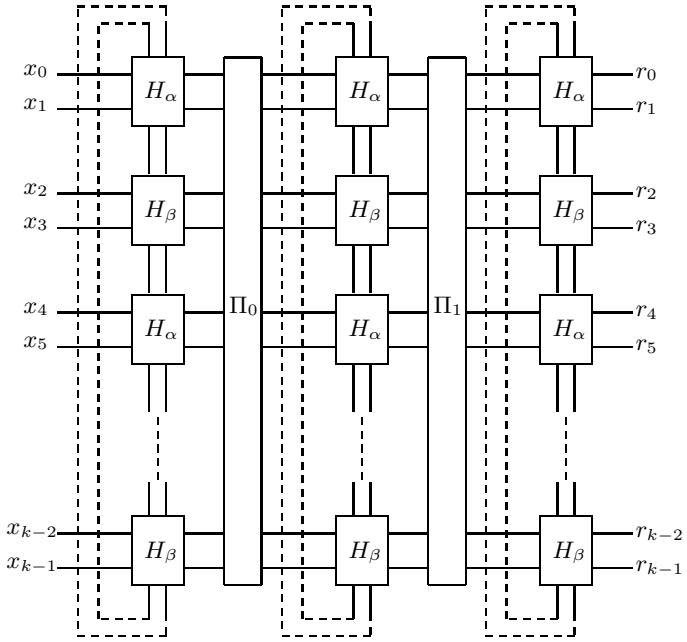


FIG. 5.2 – Turbo code série $[2k, k, d_{min}]$

Remarque 5.1.1 *L'encodage à travers un étage est réalisé en positionnant les bits d'information sur le treillis cycliques et en testant chacun des 4 états initiaux possibles pour le treillis. Celui qui permet d'obtenir un état final identique à celui de départ correspond à un chemin valide et est donc associé à un mot de code. L'encodage peut également s'effectuer à l'aide la matrice génératrice systématique associée au treillis cyclique.*

La structure de la figure 5.3 présente une construction hybride entre celle de la figure 5.2 et les codes cortex puisque le treillis cyclique de l'étage 1 est remplacé par une simple concaténation parallèle de code de Hamming étendu [8,4,4].

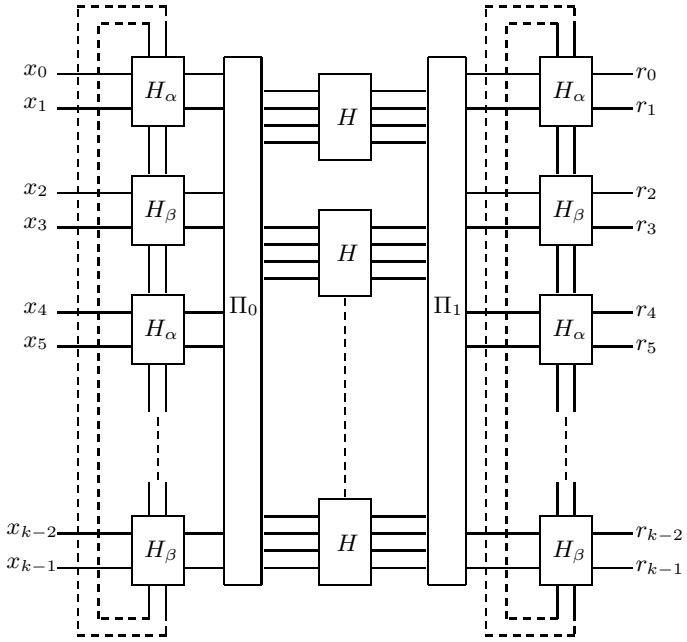


FIG. 5.3 – Turbo code série "hybride" $[2k, k, d_{min}]$

5.1.3 Codes obtenus

Avant d'énumérer les codes obtenus à partir des deux constructions précédentes on rappelle que ces codes ne nécessitent l'utilisation que de trois étages successifs de codes, ce qui est l'un des intérêts majeurs de cette construction en comparaison des codes cortex qui sont limités en terme de distance minimale dans ce cas.

Proposition 5.1.1 *Tout code cortex composé de au plus 3 étages possède une distance minimale bornée supérieurement par 12.*

Démonstration : On considère un vecteur d'information v_0 de poids 3. Nécessairement après passage du premier étage le vecteur v_1 obtenu est de poids 1. Par conséquent quelque soit la permutation utilisée pour atteindre le deuxième étage, le poids du vecteur v_2 obtenu après passage du second étage sera 3. Dans le cas "idéal", les 3 bits non-nuls sont alors répartis sur 3 codes de Hamming distincts *via* la seconde permutation et l'on obtient ainsi un vecteur v_3 de poids 9 ($= 3 \times 3$) aprés passage du troisième et dernier étage. Le mot de code ainsi construit, résultant de la concaténation des vecteurs v_0 et v_3, est donc de poids 12 ($= 3 + 9$). Quelque soit le code construit, ce cas de figure existe toujours d'où la proposition. ∎

Le tableau 5.2 présente les codes construits à partir de la construction initiale des turbo codes série (cf. figure 5.2) alors que le tableau 5.3 correspond aux codes construits à partir de la structure "hybride" (cf. figure 5.3).

code de paramètres $[n, k, d]$	nombre de permutations	couple (a, b) pour chacune des permutations	composition du treillis de base
[24,12,6]	2	(5,0) ; (7,0)	$(H_\delta \vert H_\gamma \vert H_\delta)^2$
[32,16,8]	2	(3,0) ; (3,0)	$(H_\alpha \vert H_\beta)^4$
[48,24,8]	2	(5,0) ; (11,0)	$(H_\delta \vert H_\gamma \vert H_\delta)^4$
[64,32,10]	2	(19,0) ; (19,0)	$(H_\alpha \vert H_\beta)^8$
[96,48,14]	2	(5,0) ; (11,0)	$(H_\delta \vert H_\gamma \vert H_\delta)^8$
[128,64,16]	2	(19,0) ; (11,0)	$(H_\alpha \vert H_\beta)^{16}$
[256,128,26 $\preceq d \preceq$ 32]	2	(19,0) ; (19,0)	$(H_\alpha \vert H_\beta)^{32}$

TAB. 5.2 – Codes auto-duaux de Type I

code de paramètres $[n, k, d]$	nombre de permutations	couple (a, b) pour chacune des permutations	composition du treillis de base
[32,16,8]	2	(3,0) ; (3,0)	$(H_\alpha \vert H_\beta)^4$
[64,32,12]	2	(19,0) ; (19,0)	$(H_\alpha \vert H_\beta)^8$

TAB. 5.3 – Codes auto-duaux de Type II

5.2 Polynôme d'approximation des poids

5.2.1 Principe

Cette partie présente des résultats sur la distribution moyenne des codes construits selon le modèle de la figure 5.2. La méthode de calcul est issue notamment de [9] et [8].
Considérons l'exemple du code [32,16,8]. Après l'étage 0, le polynôme énumérateur des poids est :

$$\begin{aligned} A_0(W) &= 1 + 8W^4 + 64W^6 + 316W^8 + 1856W^{10} + 6840W^{12} + 14464W^{14} + 18438W^{16} \\ &\quad + 14464W^{18} + 6840W^{20} + 1856W^{22} + 316W^{24} + 64W^{26} + 8W^{28} + W^{32} \end{aligned}$$

Après l'étage 1 nous avons :

$$\begin{aligned} A_1(W) &= 1 + 32W^6 + 300W^8 + 1952W^{10} + 6976W^{12} + 14400W^{14} + 18214W^{16} \\ &\quad + 14400W^{18} + 6976W^{20} + 1952W^{22} + 300W^{24} + 32W^{26} + W^{32} \end{aligned}$$

Et finalement, après l'étage 3 nous obtenons :

$$\begin{aligned} A_2(W) &= 1 + 364W^8 + 2048W^{10} + 6720W^{12} + 14336W^{14} + 18598W^{16} \\ &\quad + 14336W^{18} + 6720W^{20} + 2048W^{22} + 364W^{24} + W^{32} \end{aligned}$$

Dans le domaine des turbo codes, ce processus est parfois appelé "spectral thinning". Plus l'on augmente le nombre d'étages et plus la distance minimale du code peut être importante tout en diminuant le nombre de mots de code de poids faibles voisins. Cependant augmenter de façon conséquente le nombre d'étages ne permet pas nécessairement d'augmenter la distance minimale du code qui finit par tendre vers une constante.
Le calcul effectué dans l'exemple du code [32,16,8] est un calcul exact car la dimension du code est suffisamment petite pour le permettre. Cependant pour des dimensions importantes, il est possible de calculer une distribution moyenne des poids de l'ensemble des codes $[n = 2k, k]$ en moyennant sur tous les entrelaceurs possibles. On peut alors affirmer qu'il existe un code avec une meilleure distribution des poids que le code associé à la distribution moyenne obtenue.
Pour cela, on remplace l'entrelaceur réel par un entrelaceur abstrait qui est en fait une variable aléatoire ayant une distribution de probabilité uniforme. Un entrelaceur abstrait de taille k est une fonction aléatoire qui transforme un mot d'entré de poids z en l'un des quelconques et equiprobables mot de sortie permuté de même poids qui sont au nombre de $\binom{k}{z}$. Ces entrelaceurs abstraits uniformes permettent de chaîner les calculs des distributions

moyennes de poids d'un étage à l'autre. La fonction énumératrice des poids entre l'entrée et la sortie ("Input-Redundancy Weight Enumerator Function"ou encore $IRWEF$) d'un code en bloc $[n, k]$ est défini par :

$$A^C(W,Z) = \sum_{w=0}^{k}\sum_{z=0}^{k} A^C_{w,z} W^w Z^z = \sum_{w=0}^{k} A^C(w,Z) W^w = \sum_{z=0}^{k} A^C(W,z) Z^z$$

$A^C(w,Z)$ et $A^C(W,z)$ sont appelés fonction énumératrice des poids de sortie conditionnellement à l'entrée ("Conditional Weight Enumerator Function") et nous avons les deux relations :

$$A^C(w,Z) = \sum_{z=0}^{k} A^C_{w,z} Z^z$$

$$A^C(W,z) = \sum_{w=0}^{k} A^C_{w,z} W^w$$

$A^C_{w,z}$ est le nombre de mots de code de redondance de poids z générés par un mot d'information de poids w et $A(w,Z)$ est le polynôme énumérateur de la distribution des poids de redondance des mots de code générés par les mots d'information de poids w.
Pour illustrer ces notations on considère l'exemple où chaque étage est la simple concaténation de codes de Hamming étendu [8,4,4] pour construire un code $[n, k]$ comme c'est le cas pour les codes cortex. Alors l'$IRWEF$ d'un étage entier correspondant à un code C_i est :

$$A^{C_i}(W, Z_{i+1}) = (1 + 4WZ^3_{i+1} + 6W^2Z^2_{i+1} + 4W^3Z_{i+1} + W^4Z^4_{i+1})^{\frac{k}{4}}$$

Pour l'étage 0, nous obtenons :

$$A^{C_0}(W, Z_1 = W) = (1 + 14W^4 + W^8)^{\frac{k}{4}}$$

À la sortie de l'étage 1, nous obtenons un code qui est la concaténation série de C_0 (code associé à l'étage 0) et C_1 (code associé à l'étage 1). Nous pouvons alors écrire :

$$A^{C_0C_1}(W, Z_2) = \sum_{z_1=0}^{k} \frac{A^{C_0}(W, z_1)A^{C_1}(z_1, Z_2)}{\binom{k}{z_1}}$$

Après l'étage 2, nous avons :

$$A^{C_0C_1C_2}(W, Z_3) = \sum_{z_2=0}^{k}\sum_{z_1=0}^{k} \frac{A^{C_0}(W, z_1)A^{C_1}(z_1, z_2)A^{C_2}(z_2, Z_3)}{\binom{k}{z_1}\binom{k}{z_2}}$$

Plus généralement, pour l'étage s, nous avons :

$$A^{C_0C_1C_2\ldots C_s}(W, Z_{s+1}) = \sum_{z_s=0}^{k} \cdots \sum_{z_2=0}^{k} \sum_{z_1=0}^{k} \frac{A^{C_0}(W, z_1)A^{C_1}(z_1, z_2)\ldots A^{C_s}(z_s, Z_{s+1})}{\binom{k}{z_1}\binom{k}{z_2}\ldots\binom{k}{z_s}}$$

La distribution moyenne des poids à l'étage $(s+1)$ peut donc se calculer de façon itérative telle que :

$$A^{C_0\ldots C_s}(W, Z_{s+1}) = \sum_{z_s=0}^{k} \frac{A^{C_0\ldots C_{s-1}}(W, z_s)A^{C_s}(z_s, Z_{s+1})}{\binom{k}{z_s}}$$

Dans cette dernière expression, chaque entrelaceur améliore légèrement la distribution des poids car le nombre de mots de code de poids z est divisé par le coefficient binomial $\binom{k}{z}$.Chaque étage délivre une meilleure distribution à l'étage suivant, mais le gain devient vite négligeable lorsque le nombre d'étage augmente.
On donne maintenant quelques exemples de ces distributions moyennes obtenus dans deux cas distincts

- ◇ Le treillis cyclique de base est construit par une succession de blocs de sections de type $(H_\alpha|H_\beta)$.
  On donne respectivement les distributions moyennes pour des codes de paramètres [64,32], [128,64] et [256,128].

$$\begin{aligned} A_2(W) &= 1 + 2W^8 + 72W^{10} + 1542W^{12} + 22351W^{14} + \ldots \\ A_2(W) &= 1 + 13W^{16} + 430W^{18} + 13162W^{20} + 326720W^{22} + \ldots \\ A_2(W) &= 1 + W^{24} + 4W^{26} + 12W^{28} + 46W^{30} + 496W^{32} + \ldots \end{aligned}$$

- ◇ Le treillis cyclique de base est construit par une succession de blocs de sections de type $(H_\delta|H_\gamma|H_\delta)$.
  On donne respectivement les distributions moyennes pour des codes de paramètres [96,48], [120,60] et [168,84].

$$\begin{aligned} A_2(W) &= 1 + 5W^{12} + 179W^{14} + 4797W^{16} + 98068W^{18} + \ldots \\ A_2(W) &= 1 + 2W^{14} + 61W^{16} + 1949W^{18} + 51878W^{20} + \ldots \\ A_2(W) &= 1 + 2W^{18} + 10W^{20} + 228W^{22} + 7851W^{24} + \ldots \end{aligned}$$

5.2.2 Comparaison avec les codes cortex

L'intérêt premier de la construction de la figure 5.1 est qu'elle permet d'augmenter la distance du code parallèlement à l'accroissement de la longueur du code sans pour autant devoir augmenter le nombre d'étages comme

c'est le cas pour les codes cortex. Pour illustrer ce phénomène, on considère un code de paramètres [128,64] et l'on compare les distributions moyennes des codes obtenus selon le nombre de couches utilisées.

◇ Construction avec les treillis cycliques à 4 états

$$\begin{aligned} A_0(W) &= 1 + 32W^4 + 256W^6 + 1520W^8 + 11520W^{10} \dots \\ A_1(W) &= 1 + W^6 + 6W^8 + 23W^{10} + 102W^{12} + 480W^{14} \dots \\ A_2(W) &= 1 + 13W^{16} + 430W^{18} + 13162W^{20} + 326720W^{22} + \dots \end{aligned}$$

◇ Construction cortex

$$\begin{aligned} A_0(W) &= 1 + 224W^4 + 23536W^8 + 1540000W^{12} + \dots \\ A_1(W) &= 1 + 5W^4 + 86W^6 + 317W^8 + 1683W^{10} + \dots \\ A_2(W) &= 1 + 32W^8 + 493W^{12} + 8916W^{16} + \dots \\ A_3(W) &= 1 + W^8 + 3W^{10} + 8W^{12} + 34W^{14} + 98W^{16} + \dots \\ A_4(W) &= 1 + W^{12} + 26W^{16} + 26236W^{20} + \dots \\ A_5(W) &= 1 + 10W^{16} + 412W^{18} + 12979W^{20} + \dots \end{aligned}$$

Pour atteindre la distance minimale 16, la construction cortex nécessite 6 étages tandis que la nouvelle construction n'en demande que 3. De plus, au delà de 3 étages, le gain de distance minimale avec la nouvelle construction n'est pas significatif et il est donc préférable de se limiter à 3 étages quelque soit la longueur du code envisagé. En revanche pour les codes cortex, il est nécessaire d'augmenter ce nombre de couches si l'on veut un accroissement significatif de la distance minimale. Ce problème de nombre d'étages en hausse constante est d'ailleurs à l'origine des difficultés rencontrés pour décoder cette famille de code. Si la limitation à 3 étages n'assure pas nécessairement que le décodage sera aisé ou efficace, cela donne en revanche un nouvel espoir de parvenir à décoder de tels turbo codes série.

5.3 Auto-dualité

Les codes construits par la méthode présentée précédemment ont la particularité d'être des codes auto-duaux. Pour prouver ce résultat, on énonce et on démontre deux propositions dans cette section. Il est à noter que la démonstration est réalisée dans le cas de la construction n'utilisant que des treillis cycliques. Le raisonnement demeure cependant identique dans le cas de la construction "hybride".

Proposition 5.3.1 *Soient $\mathcal{C}$ un code binaire auto-dual de paramètres $[n, k, d]$ et $\mathcal{T}$ la section de treillis construite en prenant les k bits d'information en étiquettes, les $\frac{k}{2}$ premiers bits de redondance comme état entrant et*

les $\frac{k}{2}$ derniers bits de redondance comme état sortant. Alors, lorsqu'il existe, le code associé à la concaténation de ces sections sous forme d'un treillis cyclique est un code auto-dual.

Démonstration : Soit $\mathcal{T}$ un treillis cyclique, associé à un code $\mathcal{C}$, composé de s sections construites à partir d'un code auto-dual $\mathcal{C}_b$ sur le modèle de la proposition 2. Pour la $i^{ème}$ section de treillis, on note X_{i-1} le vecteur des bits d'étiquettes, α_{i-1} celui des bits d'état entrant et α_i celui des bits d'état sortant.
Pout tout i appartenant à $0, ..., s-1$, $(X_{i-1}, \alpha_i, \alpha_{i+1})$ est donc un mot du code $\mathcal{C}_b$. Alors,

$$\forall (i,j) \in (0, ..., s-1)^2, < (X_{i-1}, \alpha_i, \alpha_{i+1}), (X_{j-1}, \alpha_j, \alpha_{j+1}) >= 0 \qquad (5.1)$$

Soient $X = (X_0, ..., X_s)$ et $Y = (Y_0, ..., Y_s)$ deux mots de codes quelconques de $\mathcal{C}$. Alors, d'après (5.1), nous avons :

$$\forall (i,j) \in (0, ..., s-1)^2, < (X_i, \alpha_i, \alpha_{i+1}), (Y_j, \beta_j, \beta_{j+1}) >= 0$$

où $(X_i, \alpha_i, \alpha_{i+1})$ et $(Y_j, \beta_j, \beta_{j+1})$ sont des mots de code de $\mathcal{C}_b$. Il s'en suit

$$\forall i \in (0, ..., s-1), < X_i, Y_i > + < \alpha_i, \beta_i > + < \alpha_{i+1}, \beta_{i+1} >= 0$$

et donc

$$\sum_{i=0}^{s-1} (< X_i, Y_i > + < \alpha_i, \beta_i > + < \alpha_{i+1}, \beta_{i+1} >) = 0 \qquad (5.2)$$

Le treillis étant cyclique, cela signifie que $\alpha_0 = \alpha_s$ et $\beta_0 = \beta_s$. L'égalité (5.2) peut donc s'écrire

$$\sum_{i=0}^{s-1} (< X_i, Y_i > + 2 < \alpha_i, \beta_i >) = 0$$

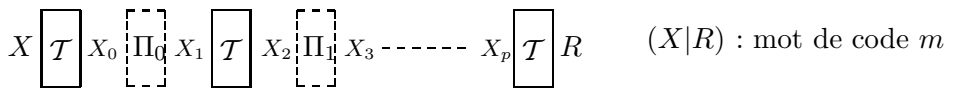
Finalement, comme nous sommes en caractéristique 2, il s'en suit

$$\sum_{i=0}^{s-1} < X_i, Y_i >= 0$$

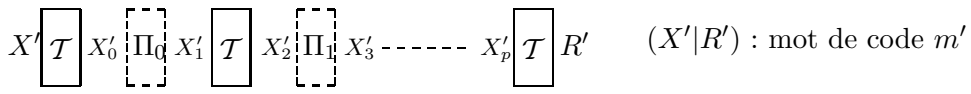
Le code $\mathcal{C}$ est donc bien un code auto-dual. ∎

Proposition 5.3.2 *Soit $\mathcal{T}$ un treillis cyclique associé à un code binaire $\mathcal{C}$ auto-dual. Alors les codes associés à la concaténation série de ces treillis via des permutations quelconques sont des codes eux aussi auto-duaux.*

Démonstration : On considère la concaténation série, *via* des permutations quelconques notées π_i $(i \in 0, ..., t-1)$, de t treillis cycliques associés à des codes auto-duaux (on note $R_{\mathcal{T}}$ la partie redondante de la matrice génératrice de ces codes). Soient $m = (X|R)$ et $m' = (X'|R')$ deux mots de code. Alors, $R = XR_{\mathcal{T}}\pi_0 R_{\mathcal{T}} \ldots \pi_{t-1}R_{\mathcal{T}}$ et $R' = X'R_{\mathcal{T}}\pi_0 R_{\mathcal{T}} \ldots \pi_{t-1}R_{\mathcal{T}}$. Chacun de ces mots peut également se représenter graphiquement de la façon suivante :



$(X|R)$: mot de code m



$(X'|R')$: mot de code m'

Avec ces notations, il nous faut donc montrer que $< m, m' >= 0$.
Chaque treillis cyclique étant associé à un code auto-dual, nous pouvons écrire :

$$(\mathcal{S}) \begin{cases} < X, X' > & + & < X_0, X'_0 > & = & 0 \\ < X_1, X'_1 > & + & < X_2, X'_2 > & = & 0 \\ \vdots & & \vdots & & \vdots \\ < X_p, X'_p > & + & < R, R' > & = & 0 \end{cases}$$

De plus, pour tout i pair, nous avons $X_{i+1} = \Pi_{\frac{i}{2}}(X_i)$. Or, pour tout couple (i, j), nous avons l'égalité $< \Pi_j(X_i), \Pi_j(X'_i) >=< X_i, X'_i >$.
Donc en particulier pour tout i pair, il existe un entier j tel que l'on ait :

$$< X_{i+1}, X'_{i+1} > \quad = \quad < \Pi_j(X_i), \Pi_j(X'_i) > \quad = \quad < X_i, X'i >$$

Le système $(\mathcal{S})$ devient alors :

$$\begin{cases} < X, X' > & + & < X_0, X'_0 > & = & 0 \\ < X_0, X'_0 > & + & < X_2, X'_2 > & = & 0 \\ < X_2, X'_2 > & + & < X_4, X'_4 > & = & 0 \\ \vdots & & \vdots & & \vdots \\ < X_p, X'_p > & + & < R, R' > & = & 0 \end{cases}$$

Si l'on somme chacune des équations du système, nous obtenons l'égalité suivante :

$$< X, X' > + < R, R' > + 2 \sum_{i\ pair} < X_i, X'_i >= 0$$

Les opérations étant réalisées en caractéristique 2, nous avons donc

$$< X, X' > + < R, R' >= 0$$

Par conséquent,

$$< m, m' > \ = \ < (X|R), (X'|R') > \ = \ < X, X' > + < R, R' > \ = \ 0$$

Les codes obtenus par la concaténation série sont donc bien auto-duaux. ■

Remarque 5.3.1 *Soient $\mathcal{C}$ un code auto-dual et $\mathcal{T}$ le treillis cyclique qui lui est associé. Alors les sections de treillis de $\mathcal{T}$ ne sont pas nécessairement associées à des codes auto-duaux.*

5.4 Décodage

L'une des difficultés majeures du décodage des codes "cortex" est le positionnement sur la structure des bits composant les mots de code. Ces bits se trouvent en effet uniquement sur les bords de la structure et deviennent rapidement minoritaire en nombre par rapport aux bits positionnés sur les étages internes de la structure et qui ne sont pas transmis sur le canal lorsque l'on souhaite construire un code possédant une bonne distance minimale. En fait, ce phénomène apparaît dès que l'on utilise plus de 3 étages dans la construction "cortex". Lors du décodage, il y a donc sur la structure plus de bits ne recevant aucune estimation du canal que de bits en possédant une. De ce fait il est très difficile de permettre au décodeur de démarrer dans de bonnes conditions car il doit dans un premier temps estimer ces bits n'ayant reçu aucune information en provenance du canal de transmission.
Malheureusement, les codes "cortex", s'ils n'utilisent que 3 étages, ne peuvent pas atteindre une distance minimale supérieures à 12. Tout code "cortex" de distance minimale supérieure à 12 nécessite l'utilisation de plus de 3 étages et le nombre de bits transmis sur le canal devient alors minoritaire par rapport aux bits non transmis et donc non estimés à la réception du message. Le décodage de ces codes est donc problématique pour les distances strictement supérieure à 12, ce qui les rend difficilement utilisables

dans la pratique.
La nouvelle construction série présentée dans ce chapitre s"affranchit de ce problème lié au nombre d'étages car les distances minimales des codes construits ne sont pas bornées si on se limite à 3 étages en raison des propriétés bien singulières des polynômes de transition des poids des treillis cycliques utilisés. Par conséquent, quelque soit le code construit, il y a toujours sur la structure autant de bits transmis sur le canal que de bits non transmis. Pour réaliser le décodage, nous avons donc autant de bits possédant une estimation provenant du canal de transmission que de bits n'en possédant pas. Cependant, cet équilibre entre les bits estimés et ceux qui ne le sont pas ne permet pas d'assurer que le décodage sera performant.
Pour des raisons de temps, les applications pratiques concernant le décodage n'ont malheureusement pas pu être mise en œuvre durant cette thèse. La question de la qualité de ces codes dans le cadre d'une utilisation pratique reste donc pleinement à vérifier.

Conclusion

Cette thèse a été consacrée à l'étude du codage correcteur d'erreurs et plus particulièrement à la famille des turbo codes. Après un rappel sur les différentes techniques de codage que l'on peut rencontrer dans la littérature, nous avons considéré l'aspect plus applicatif du domaine avec la présentation de plusieurs méthodes de décodage numériques et analogiques. Nous avons vu que ces méthodes sont très fortement liées à la structure graphique utilisée pour représenter le code et que ces structures sont très diverses pour un même code et qu'il est donc très difficile de trouver la mieux adaptée au décodage. Nous avons également vu qu'il était possible de décoder sans le moindre support graphique mais que ces méthodes avaient l'inconvénient d'être très limitées car elles deviennent très rapidement trop complexes à mettre en œuvre lorsque les codes atteignent des dimensions significatives.

L'un des buts de cette thèse était de sélectionner l'algorithme de décodage le plus approprié et de l'adapter aux codes cortex qui sont des codes possédant d'excellentes distances minimales mais dont le décodage est particulièrement ardu. Dans le troisième chapitre, nous nous sommes particulièrement attardés sur ce problème. Nous sommes parvenus à décoder notamment les codes cortex composés de trois couches et de distance minimale 8 en mettant en avant le lien qu'il existe entre ces codes et les treillis cycliques. Plus précisément, il est possible de déplier la structure cortex pour faire apparaître un treillis bracelet comportant 16 états. De ce fait, il est possible de les décoder de façon optimale. Un résultat similaire existe si l'on utilise $\mathbb{Z}_4$ comme alphabet et l'octacode comme code composant. Dans ce cas, les codes cortex composés de trois couches et de distance minimale 12 peuvent se représenter sous la forme de treillis bracelet à 256 états. Cependant une majorité des codes cortex n'ont pu être décodés pour une raison relativement simple : ces codes sont composés de plusieurs couches assemblées en série, mais seuls les bits présents sur les bords de la structure

sont transmis sur le canal. Par conséquent lors de la première itération du décodage toute l'information reçue est concentrée sur les bords de la structure alors que le centre est dépourvu de toute information provenant du canal. Le centre comportant plus de bits que les bords, cela explique partiellement la difficulté de décoder cette famille de codes. Pour remédier en partie à ce problème, nous avons proposée une évolution de ces codes en répartissant les bits à transmettre sur l'ensemble de la structure. Ainsi chaque couche de la structure comporte des bits qui seront émis et d'autres non. Si de prime abord nous avons placé beaucoup d'espoir dans cette méthode car elle permettait de mieux recouvrir la structure avec les bits du canal tout en préservant les bonnes propriétés de distance minimale, le décodage s'est là aussi soldé par un échec. Cependant, dans cette nouvelle configuration, le nombre de bits émis reste insuffisant en comparaison de ceux qui ne le sont pas.

Dans le quatrième chapitre, devant l'impossibilté de décoder les codes cortex, nous avons donc décidé de mieux comprendre les critères de sélection pour les codes composants d'un turbo code et de le mettre en pratique pour la construction de turbo codes à concaténation parallèle multiple. Le fait marquant de cette étude est que la distance minimale du code composant n'est pas le critère primordial, mais qu'il est au contraire préférable de privilégier des codes possédant de bons polynômes de transition des poids. L'aspect intéressant de cette méthode de sélection des codes de base est qu'elle permet de prendre des codes très peu complexes dès l'instant où l'aspect distance minimale est délaissé (la complexité d'un treillis est très fortement liée à la distance minimale du code qui lui est associé) au profit d'un nouveau critère. Ainsi, nous avons sélectionné des treillis cycliques construits à partir du code de Hamming étendu [8,4,4] qui ne possèdent que 4 états mais qui présentent les propriétés espérées concernant leurs polynômes de transition des poids. En utilisant ces treillis nous avons obtenu des turbo codes courts présentant de bonnes distances minimales et nous sommes parvenus à retrouver certains codes auto-duaux extrémaux. Par exemple, pour la première fois le code de Golay [24,12,8] a été représenté sous une forme de turbo code comme étant la concaténation parallèle de trois treillis cycliques à 4 états. Seules les constructions utilisant une concaténation simple de deux treillis ont déjà été décodée, et bien que ce ne sont pas celles qui fournissentt les meilleures distances minimales, les résultats sont très encourageant notamment pour les codes courts comme par exemple un [400,200] dont les performances sont similaires à une tubo code de même longueur mais utilisant des treillis à 16 états.

Cette construction est toutefois “fragilisée” car elle utilise un procédé de concaténation multiple et par conséquent il faut se limiter à l'utilisation de quatre voire cinq treillis au maximum pour ne pas trop handicaper le décodage. Une question ouverte est également posée dans ce chapitre. Les codes cortex sont en effet une famille de codes dont la partie redondante des matrices génératices est circulante par bloc si les permutations utilisées sont linéaires. Avec cette hypothèse cela signifie que les codes cortex peuvent se mettre sous la forme d'un turbo code à concaténation multiple. L'aspect intéressant est que les codes que l'on concatène possèdent de faibles distances minimales et laisse espérer que leur repésentation en treillis soit réalisable avec une complexité raisonnable. Cela permettrait ainsi de décoder les codes cortex en utilisant les méthodes itératives classiques et nous aurions à portée de main une famille de turbo codes avec de très bonnes distances minimales et un décodage peu complexe à mettre en œuvre.

Dans le cinquième et dernier chapitre, nous avons réalisé une évolution des codes cortex en utilisant les enseignements acquis sur les critères de choix des codes composants. En effet les différentes couches des codes cortex sont composées d'une simple concaténation de petits codes et ne permet pas de respecter les critères fixés concernant le polynôme de transition de poids. Nous avons donc ajouté des connexions verticales pour relier les différents petits codes d'une même couche. Cet ajout de connexions revient en fait à considérer que les couches de codes sont des treillis cycliques à 4 états très semblables à ceux du chapitre précédent car eux aussi élaborés à partir du code de Hamming étendu [8,4,4]. Les codes obtenus par cette méthode sont comme pour les codes cortex des codes auto-duaux. L'intérêt d'une telle évolution de la structure est qu'elle permet dorénavant d'obtenir des codes avec de bonnes distances minimales sans avoir pour autant à utiliser plus de trois couches (les codes cortex ont une distance minimale bornée à 12 si l'on considère au plus trois couches). Ainsi il y a autant de bits sur les bords de la structure (bits émis sur le canal) que de bits au centre de la structure (bits non émis si l'on désire un rendement supérieur à $\frac{1}{2}$) et l'on compense en partie l'une des difficultés majeures du décodage des codes cortex. Cependant le fait que seul la moitié des bits soient transmis sur le canal risque de compromettre les performances de décodage qui seront toutefois meilleures que pour les codes cortex.

Les turbo codes sont maintenant devenus incontournables dans les systèmes de transmission numérique et leur utilisation est préconisée dans de plus en plus de normes (UMTS, BRAN, ...). Leur dernier véritable

handicap demeure le phénomène “d’error floor” qui est en grande partie due à leur distance minimale encore trop faible, et une complexité que l’on souhaite sans cesse être en mesure de diminuer. Cependant il apparaît qu’il est possible d’obtenir des turbo codes présentant de très bonnes distances minimales et qu’il est également possible de représenter certains codes auto-duaux extrémaux sous cette forme. Si le phénomène “d’erreur floor” ne peut pas être totalement supprimé, l’espoir demeure donc de le repousser de façon très significative en cherchant à élaborer des turbo codes avec de meilleures distances minimales. Il apparaît également que la complexité pourrait être très praticable car nous avons pu élaborer des turbo codes utilisant des treillis à seulement 4 états. Un dernier aspect à ne pas perdre de vue concerne les symétries dans les structures ainsi que la simplicité de représentation des treillis utilisés. Prendre en compte ces deux derniers critères permet d’envisager la réalisation de décodeurs analogiques qui ont l’avantage de ne pas subir le coût de complexité dû aux nombreuses itérations lors du décodage.

Annexe A

Critères de décodage

Selon que les observations sont binaires ou réelles, les algorithmes de décodage mis en œuvre ne sont pas les mêmes. Aussi nous formulons quelques rappels sur les critères de décodage suivant que l'on cosidère le décodage ferme (dur ou "hard") qui utilise des entrées binaires résultant du seuillage de la sortie du canal ou le décodage pondéré (souple ou "soft") qui utilise des entrées réelles. On fait également l'hypothèse que le décodage ferme est réalisé sur un canal sans mémoire (on peut ainsi se ramener à un modèle de canal binaire symétrique) et le décodage pondéré sur un canal gaussien.

A.1 Décodage ferme

On suppose que la probabilité qu'une erreur se produise est égale à p. On considère alors un code binaire $\mathcal{C}$ de longueur n. On note $c = (c_0, c_1, \ldots, c_{n-1})$ un mot de code et $r = (r_0, r_1, \ldots, r_{n-1})$ l'observation qui lui est associée. Le décodeur cherche à maximiser la vraisemblance *a posteriori* donc la probabilité $P(c|r)$. Les mots de code étant équiprobables, cela revient par conséquent à maximiser $P(r|c)$. L'hypothèse d'un canal sans mémoire permet de dire que les observations r_i sont indépendantes et par conséquent :

$$P(r|c) = \prod_{i=0}^{n} P(r_i|c_i)$$

De plus selon qu'il y ait égalité ou non entre r_i et c_i, $P(r_i|c_i)$ prend la valeur $(1-p)$ ou p. Nous pouvons donc écrire :

$$P(r|c) = p^{d(r,c)} \times (1-p)^{n-d(r,c)}$$

où $d(.,.)$ désigne la distance de Hamming.
p étant inférieure à $\frac{1}{2}$, maximiser $P(r|c)$ revient donc à minimiser la distance de Hamming entre c et r. Le décodage ferme consiste donc à rechercher le mot de code c de $\mathcal{C}$ qui minimise la distance de Hamming avec l'observation r.

A.2 Décodage souple

Deux aspects sont ici à envisager. On peut considérer une minimisation de la probabilité d'erreur par mot (on maximise la vraisemblance *a posteriori*) ou une minimisation de la probabilité d'erreur par bit (on maximise le rapport de vrasemblance *a posteriori*).

A.2.1 Probabilité d'erreur par mot

On note $c = (c_0, c_1, \ldots, c_{n-1})$ un mot de longueur n du code $\mathcal{C}$ et $r = (r_0, r_1, \ldots, r_{n-1})$ l'observation qui lui est associée. On cherche à maximiser $P(c|r)$. Les mots de codes étant equiprobables, cela équivaut à maximiser $p(r|c)$. Le canal étant gaussien, les observations sont indépendantes et nous pouvons écrire :

$$P(r|c) = \prod_{i=0}^{n} P(r_i|c_i) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{d^2(r,c)}{2\sigma^2}\right)$$

où $d(r,c)$ est la distance euclidienne entre le mot c et l'observation r.
Maximiser $P(r|c)$ revient donc à minimiser la distance euclidienne entre l'observation r et le mot c de $\mathcal{C}$. Un tel décodeur fournit donc le mot de code le plus probable.

A.2.2 Probabilité d'erreur par bit

Pour chacune des composantes c_i du mot de code, on détermine le logarithme du rapport de vraisemblance défini par :

$$L(c_i) = \frac{P(c_i = 1|r)}{P(c_i = 0|r)}$$

Si $L(c_i)$ est supérieure à 1 on décide que le bit vaut 1 et 0 dans le cas contraire.
Ce décodeur traite donc les éléments bits à bits ce qui implique que le mot obtenu après décodage n'appartient pas nécessairement au code.

Bibliographie

[1] S. Aji, G. Horn, R. McEliece, and M. Xu. Iterative min-sum decoding of tail-biting codes. ITW 1998, pages 68–69, Killarney (Ireland), June 1998.

[2] S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2) :325–343, March 2000.

[3] J.B. Anderson. Best short rate 1/2 tailbiting codes for the bit-error rate criterion. *IEEE Transactions on Information Theory*, 48(4) :597–610, April 2000.

[4] J.B. Anderson and S.M. Hladik. Tailbiting MAP decoders. *IEEE Journal on Selected Areas in Communications*, 16(2) :297–302, February 1998.

[5] J.B. Anderson and K.E. Tepe. Properties of the tailbiting BCJR decoder. In *Codes, Systems and Graphical Models*, IMA Summer Program, pages 211–238. Springer, May 1999.

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

[7] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Analysis, design and iterative decoding of double serially concatenated codes with interleavers. *IEEE Journal on Selected Areas in Communications*, 16(2) :231–244, February 1998.

[8] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes : Performance analysis, design and iterative decoding. *IEEE Transactions on Information Theory*, 44(3) :909–926, May 1998.

[9] S. Benedetto and G. Montorsi. Unveiling turbo codes : some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, IT-42(2) :409–428, March 1996.

[10] S. Benedetto and G. Montorsi. Generalized concatenated codes with interleavers. International Symposium on Turbo-Codes, pages 32–39, Brest (France), September 1997.

[11] C. Berrou, C. Douillard, and S. Kerouédan. The advantages of non-binary Turbo Codes. ITW2001, Cairns (Australia), September 2001.

[12] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : Turbo-Codes. *IEEE Transactions on Communications*, 44(10) :1261–1271, October 1996.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correction coding and decoding Turbo-Codes. volume 2 of *ICC'93 Proceedings*, pages 1064–1071, Geneva (Switzerland), May 1993.

[14] E. Biglieri, D. Divsalar, P.J. Mclane, and M.K. Simon. *Introduction to Trellis-Coded Modulation with Applications*. Macmillan Publishing Company.

[15] I.E. Bocharova, B.D. Kudryashov, R. Johannesson, and P. Ståhl. Searching for tailbiting codes with large minimum distances. ISIT2000 Proceedings, page 341, Sorrento (Italy), June 2000.

[16] E. Cadic. Étude de méthodes de décodage analogique et itératif numérique à décision douce de codes correcteurs d'erreurs. Juillet 2000.

[17] E. Cadic and J.C. Carlach. Building short turbo codes with high minimum distance using 4-state tail-biting trellises. ISIT2003 Proceedings, Yokohama (Japan), June 2003.

[18] E. Cadic and J.C. Carlach. Construction de turbo-codes à treillis à 4 états ayant de bonnes propriétés de distance. GRETSI 2003, Paris (France), Septembre 2003.

[19] E. Cadic and J.C. Carlach. Short turbo codes with high minimum distance using 4-state tail-biting trellises. 3^{rd} International Symposium on Turbo Codes, Brest (France), September 2003.

[20] E. Cadic, J.C. Carlach, G. Olocco, A. Otmani, and J.P. Tillich. Minimal tail-biting trellises of extremal self-dual codes of length 24, 32 and 40 over $GF(2)$ and $\mathbb{Z}_4$. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, AAECC-14, pages 57–66. Springer, Melbourne (Australia), November 2001.

[21] A.R. Calderbank, G.D. Forney, and A. Vardy. Minimal tail-biting trellises : The Golay code and more. *IEEE Transactions on Information Theory*, 45(5) :1435–1455, July 1999.

[22] R. Calderbank and J. Maso. A new description of trellis codes. *IEEE Transactions on Information Theory*, IT-30 :784–791, November 1984.

[23] A. Canteaut and F. Chabaud. A new algorithm for finding minimum weight words in a linear code : Application to McElieces's cryptosystem and to narrow-sence BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1) :367–378, January 1998.

[24] A. Canteaut and P. Charpin. Weight distribution properties of codes. *Private communication*, June 1998.

[25] J.C. Carlach, A. Otmani, and C. Vervoux. A new scheme for building good self-dual codes. ISIT2000 Proceedings, page 476, Sorrento (Italy), June 2000.

[26] J.C. Carlach and C. Vervoux. A new family of block turbo-codes. AAECC-13 Proceedings, pages 15–16, Honolulu (Hawaii), November 1999.

[27] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, IT-18, January 1972.

[28] J.H. Conway and N.J.A. Sloane. Soft decoding techniques for codes and lattices, including the golay code and the leech lattice. *IEEE Transactions on Information Theory*, IT-32(1) :41–50, January 1986.

[29] J.H. Conway and N.J.A. Sloane. A new upper bound on the minimal distance of self-dual codes. *IEEE Transactions on Information Theory*, 36 :1319–1333, November 1990.

[30] D. Divsalar, S. Dolinar, and F. Pollara. Low complexity turbo-like codes. Proceedings of the 2nd International Symposium on Turbo-Codes, pages 73–80, Brest (France), September 2000.

[31] D. Divsalar and F. Pollara. Serial and hybrid concatenated codes with applications. International Symposium on Turbo Codes, pages 80–87, Brest (France), 1997.

[32] I. Dumer. Concatenated codes and their multilevel generalizations. In *part II of Handbook of Coding Theory*, pages 1911–1988. North-Holland/Elsevier Publishers, 1998.

[33] I. Dumer and R. Krichevskiy. Soft-decision majority decoding of Reed-Muller codes. *IEEE Transactions on Information Theory*, 46(1) :258–264, January 2000.

[34] M. Esmaeili and A.K. Khandani. Acyclic tanner graphs and maximum-likelihood decoding of linear block codes. *IEEE Proc.-Commun.*, 147(6) :322–332, December 2000.

[35] T. Etzion, A. Trachtenberg, and A. Vardy. Which codes have cycle-free Tanner graphs ? *IEEE Transactions on Information Theory*, 45(6) :2173–2181, September 1999.

[36] P.G. Farrell. Graph decoding of error-control codes. DSPCS'99, Scarborough, Perth (Australia), February 1999.

[37] F. Fekri, W. McLaughlin, R.M. Mersereau, and R.W. Schafer. Wavelet convolutionnal codes with bipartite trellises. ISIT2001, page 245, Washington (USA), June 2001.

[38] M. Ferrari and S. Bellini. Cross-entropy, soft decoding and turbo decoding. Proceedings of the 2nd International Symposium on Turbo-Codes, pages 35–38, Brest (France), September 2000.

[39] G.D. Forney. *Concatenated Codes*. PhD thesis, Massachusets Institute of Technology (USA), 1966.

[40] G.D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61 :268–278, March 1973.

[41] G.D. Forney. Coset codes - part 2 : Binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5) :1152–1186, September 1988.

[42] G.D. Forney. On iterative decoding and the two-way algorithm. International Symposium on Turbo Codes, pages 12–25, Brest (France), 1997.

[43] G.D. Forney. Codes on graphs : Generalized state realizations. *IMA Summer Program*, August 1999.

[44] G.D. Forney. Codes on graphs : Normal realizations. *IEEE Transactions on Information Theory*, 47(2) :520–548, February 2001.

[45] G.D. Forney, F.R. Kschischang, B. Marcus, and S. Tuncel. Iterative decoding of tail-biting trellises and connections with symbolic dynamics. In *Codes, Systems and Graphical Models*, IMA Summer Program, pages 239–264. Springer, May 1999.

[46] M.P.C. Fossorier and S. Lin. Some decomposable codes : The $|a + x|b + x|a + b + x|$ construction. *IEEE Transactions on Information Theory*, 43(5) :1663–1667, September 1997.

[47] T. Fujiwara, H. Yamamoto, T. Kasami, and S. Lin. A trellis-based recursive maximum-likelihood decoding algorithm for binary linear block codes. *IEEE Transactions on Information Theory*, 44(2) :714–729, March 1998.

[48] R.G. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 8 :21–28, January 1962.

[49] A. Guyader and E. Fabre. Dealing with short cycles in graphical codes. ISIT2000 Proceedings, page 10, Sorrento (Italy), June 2000.

[50] J. Hagenauer. Iterative decoding of binary block and convolutionnal codes. *IEEE Transactions on Information Theory*, 42(2) :429–445, March 1996.

[51] J. Hagenauer and E. Offer. Joint equalization and decoding with analog feedback non-linear network. EPMCC'99, Paris(France), March 1999.

[52] J. Hagenauer and E. Offer. Analog turbo-networks in VLSI : The next step in turbo-decoding and equalization. Proceedings of the 2nd International Symposium on Turbo-Codes, pages 209–218, Brest (France), September 2000.

[53] J. Hagenauer, E. Offer, C. Measson, and M. Mörz. Decoding and equalization with analog non-linear networks. *European Transactions on Telecommunications(ETT)*, 10(6) :659–680, October 1999.

[54] J. Hagenauer and M. Winklhofer. The analog decoder. ISIT'1998, Cambridge (England), August 1998.

[55] M. Handlery and J.B. Anderson. Best tailbiting convolutionnal encoders using a priori information. ISIT2000 Proceedings, page 340, Sorrento (Italy), June 2000.

[56] A.R. Hummons, P.V. Kumar, A.R. Calderbank, N.J.A. Sloane, and P. Solé. The $\mathbb{Z}_4$ linearity of kerdock, preparata, goethals and related codes. *IEEE Transactions on Information Theory*, 40(2) :301–319, March 1994.

[57] H. Imai and S. Hirakawa. A new multilevel coding method using error-correcting codes. *IEEE Transactions on Information Theory*, IT-23(3) :371–377, May 1977.

[58] R. Johannesson, P. Ståhl, and E. Wittenmark. A note on type II convolutionnal codes. *IEEE Transactions on Information Theory*, 46(4) :1510–1515, July 2000.

[59] R. Johannesson and K.S. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE Press, 1999.

[60] N. Kamiya. On algebraic soft-decision decoding algorithms for BCH codes. *IEEE Transactions on Information Theory*, 47(1) :45–58, January 2001.

[61] T. Kasami, T. Takata, T. Fujiwara, and S. Lin. On the optimum bit orders with respect to the state complexity of trellis diagrams for binary linear codes. *IEEE Transactions on Information Theory*, 39(1) :242–245, January 1993.

[62] R. Kötter and A. Vardy. Construction of minimal tail-biting trellises. ITW 1998, pages 72–74, Killarney (Ireland), June 1998.

[63] R. Kötter and A. Vardy. The structure of tail-biting trellises : Minimality and basic principles. May 2002.

[64] S. Lin and R.Y. Shao. General stucture and construction of tail biting trellises for linear block codes. ISIT2000 Proceedings, page 117, Sorrento (Italy), June 2000.

[65] Y. Liu, S. Lin, and M.P.C. Fossorier. MAP algorithms for decoding linear block codes based on sectionalized trellis diagrams. *IEEE Transactions on Communications*, 48(4) :577–587, April 2000.

[66] H.A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy. Probability propagation and decoding in analog VLSI. ISIT'1998, Cambridge (England), August 1998.

[67] H.A. Loeliger, F. Tarköy, F. Lustenberger, and M. Helfenstein. Decoding in analog VLSI. *IEEE Communications Magazine*, pages 99–101, April 1999.

[68] H-L. Lou. Implementing the Viterbi algorithm. *IEEE Signal Processing Magazine*, pages 42–52, September 1995.

[69] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland publishing company, 1981.

[70] T. Mittelholzer. Subcode graphs of linear block codes. ISIT2000 Proceedings, page 116, Sorrento (Italy), June 2000.

[71] M. Moerz, A. Schaefer, and E. Offer. Analog decoding of high rate tailbiting codes using the dual trellis. ISIT2001, page 331, Washington (USA), June 2001.

[72] M. Mörz, J. Hagenauer, and E. Offer. Decoding and equalization with on the implementation of the app (bcjr) algorithm. ISIT2000 Proceedings, page 425, Sorrento (Italy), June 2000.

[73] G. Olocco and J.P. Tillich. Iterative decoding of a new family of block turbo-codes. 2^{nd} International Symposium on Turbo Codes and Related Topics, Brest(France), September 2000.

[74] G. Olocco and J.P. Tillich. A familly of self-dual codes which behave in many respects like random linear codes of rate $\frac{1}{2}$. ISIT2001 Proceedings, page 15, Washington DC (USA), June 2001.

[75] H. Oral. Constructing self-dual codes using graphs. *Journal of Combinatorial Theory*, pages 250–258, November 1988.

[76] A. Otmani. Recherche des codes extrémaux sous forme de codes cortex. Juin 1999.

[77] A. Otmani. Codes cortex et construction de codes auto-duaux optimaux. Faculté des sciences de Limoges, Septembre 2002.

[78] R. Pellizzoni and A. Spalvieri. Binary multilevel coset codes based on Reed-Muller codes. *IEEE Transactions on Communications*, 42(7) :2357–2360, July 1994.

[79] L. Ping and K.Y. Wu. Concatenated tree codes. Proceedings of the 2nd International Symposium on Turbo-Codes, pages 161–164, Brest (France), September 2000.

[80] L. Ping and K.L. Yeung. Symbol-by-symbol APP deoding of the Golay code and iterative decoding of concatenated Golay codes. *IEEE Transactions on Information Theory*, 45(7) :2558–2562, November 1999.

[81] A. Reznik. *Iterative Decoding of Codes Defined on Graphs*. PhD thesis, Massachusets Institute of Technology, Massachusets (USA), July 1998.

[82] P. Shankar, P.N.A Kumar, and K. Sasidharan. ML decoding of block codes on their tailbiting trellises. ISIT2001, page 291, Washington (USA), June 2001.

[83] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, pages 379–423,623–656, October 1948.

[84] R.Y. Shao, S. Lin, and M.P.C. Fossorier. Two simple stopping criteria for turbo decoding. *IEEE Transactions on Communications*, 47 :1117–1120, August 1999.

[85] J. Snyders and Y. Be'ery. Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes. *IEEE Transactions on Information Theory*, 5(35) :963–975, September 1989.

[86] U. Sorger and S. Fedorenko. The "star trellis" of the Golay code. Seventh International Workshop on Algebraic and Combinatorial Coding Theory, pages 288–292, Bansko (Bulgaria), June 2000.

[87] R.M. Tanner. A recursive approach to low-complexity codes. *IEEE Transactions on Information Theory*, IT-27 :533–547, September 1981.

[88] R.M. Tanner. The tetrahedral golay code. April 2000.

[89] G. Ungerboeck. Treillis-coded modulation with redundant signal sets part 1 : Introduction. *IEEE Communications Magazine*, 25(2) :5–11, February 1987.

[90] G. Ungerboeck. Treillis-coded modulation with redundant signal sets part 2 : State of the art. *IEEE Communications Magazine*, 25(2) :12–21, February 1987.

[91] A. Vardy. The Nordstrom-Robinson code : Representation over $GR(4)$ and efficient decoding. *IEEE Transactions on Information Theory*, 40(5) :1686–1693, September 1994.

[92] A. Vardy and Y. Be'ery. More efficient soft decoding of the Golay codes. *IEEE Transactions on Information Theory*, 37(3) :667–672, May 1991.

[93] S. Vialle and J. Boutros. A Gallager-Tanner construction based on convolutional codes. WCC'99, pages 393–403, Paris (France), Janvier 1999.

[94] A.J. Viterbi. Error bounds for convolutionnal codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2) :260–269, April 1967.

[95] Q. Wang and L. Wei. Graph-based iterative decoding algorithms for parity-concatenated treillis codes. *IEEE Transactions on Information Theory*, 47(3) :1062–1074, March 2001.

[96] Y-P. Wang, R. Ramésh, A. Hassan, and H. Koorapaty. On MAP decoding for tail-biting convolutional codes. ISIT, page 225, Ulm (Germany), June-July 1997.

[97] P. Weiner. *Basic Properties of Multidimensional Convolutionnal Codes*, volume Codes, Systems, and Graphical Models, pages 397–414. Springer, 1999.

[98] C. Weiß, C. Bettstetter, and S. Riedel. Code constuction and decoding of parallel concatenated tail-biting codes. *IEEE Transactions on Information Theory*, 47(1) :366–386, January 2001.

[99] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Linköping (Sweden), April 1996.

[100] N. Wiberg, H.-A. Loeliger, and R. Kötter. Codes and iterative decoding on general graphs. volume 6 of *European Transactions on Telecommunications*, pages 513–525, Sept-Oct 1995.

[101] J.K. Wolf. Efficient maximum likelihood decoding of linear block codes. *IEEE Transactions on Information Theory*, IT-24(1) :79–80, January 1978.