Reconnaissance de codes correcteurs

Audrey Tixier

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Audrey TIXIER

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Reconnaissance de codes correcteurs

soutenue le 14 octobre 2015

devant le jury composé de :

Jean-Pierre Tillich	Inria Paris-Rocquencourt	Directeur de thèse
Roland Gautier	Université de Bretagne Occidentale	Rapporteur
Pierre Loidreau	Direction Générale de l'Armement	Rapporteur
Thierry Berger	Université de Limoges	Examinateur
Claude Berrou	Télécom Bretagne	Examinateur
Nicolas Sendrier	Inria Paris-Rocquencourt	Examinateur
Annick Valibouze	Université Pierre et Marie Curie	Examinatrice

Audrey Tixier

Reconnaissance de codes correcteurs

Sous la direction de Jean-Pierre Tillich

Inria - Paris Rocquencourt
Equipe SECRET

Octobre 2015

Remerciements

Cette thèse a été effectuée au sein de l'équipe-projet Secret de l'Inria Paris-Rocquencourt. Je vais essayer de remercier dans ces quelques lignes toutes les personnes qui m'ont permis de réaliser cette thèse et qu'elle se déroule dans les meilleurs conditions. Je m'excuse d'avance au près de toutes les personnes que je vais malheureusement et involontairement oublier.

Mes premiers remerciements vont à Jean-Pierre, mon directeur de thèse. Pendant ces trois années il n'a cessé de me guider, de m'aider et de m'encourager. Je le remercie de tout cela ainsi que du temps qu'il a su me consacrer malgré son emploi du temps plus que compliqué. Merci également à Nicolas pour sa disponibilité et son soutien tout au long de ces années.

Je tiens aussi à remercier mes deux rapporteurs, Roland Gautier et Pierre Loidreau, qui ont donné de leur temps pour la relecture de ce manuscrit ainsi que Thierry Berger, Claude Berrou et Annick Valibouze qui ont accepté de faire partie de mon jury.

Je souhaite évidemment remercier Maxime, Jérôme, Florine et Marion pour leur soutien et leur disponibilité. Malgré la quantité de questions (plus ou moins intéressantes...) que je suis venue leur poser ils ont toujours été là et en particulier Maxime le principal destinataire de mes interrogations. Je remercie aussi Angélique, François, Pierre et Marc malgré leur manque de temps évident ils ont toujours suivi mes travaux.

Mes remerciements vont également vers tous les membres du projet Secret et plus particulièrement aux permanents : André, Anne, Anthony, Gaëtan, Jean-Pierre, María, Nicolas et Pascale sans qui l'ambiance du bâtiment 25 ne serait être la même. Même si ce numéro ne sera bientôt plus lié au projet Secret je suis sûre qu'ils sauront recréer cette ambiance dans la nouvelle maison Secret.
Je pense aussi à tous les étudiants et post-doctorants, que j'ai pu y côtoyer : Adrien, Aurélie, Christina, Denise, Dimitris, Grégory, Irene, Joëlle, Julia, Kaushik, Marion, Nicky, Rafael, Rodolfo, Sebastien, Valentin, Virginie et Yann.
Un grand merci à Christelle pour sa disponibilité et son efficacité. Elle m'a toujours aidé avec le sourire, et ce quel que soit le degrés d'étrangeté de mes demandes...

Je ne pourrai écrire ces quelques lignes sans remercier ceux avec qui j'ai partagé mon bureau. Il y eu tout d'abord Marion, je la remercie pour son aide précieuse dès le début de ma thèse (et même avant) mais aussi pour les différentes discussions ou réflexions communes que nous avons pu avoir. Même si l'on ne partage plus le même bureau elle est restée présente et encourageante, je l'en remercie.
Je pense, évidemment, aussi à Joëlle. Même s'il parait que j'ai perdu (de mon point de vue

je dirais ex aequo !) ces trois années n'auraient pas été les mêmes sans nos discussions sur des sujets existentiels ainsi que nos explications en mode "cobayes". Je la remercie également pour l'entre-aide pratique et psychologique lors de la rédaction ainsi que pour les démarches administratives et logistiques de ces dernières semaines.
Je remercie aussi mes deux co-bureau par intermittence ; Marià pour sa bonne humeur permanente ainsi que Denis l'irremplaçable "raleur" : la seule personne pour laquelle je m'inquiète s'il ne râle pas pendant au moins une heure !

Nous n'étions pas dans le même bureau mais je tiens également à remercier plus particulièrement Julia pour son soutien et nos conversations (généralement sur des histoires d'équations...). C'est aussi une des rares personnes avec qui j'ai pu échanger des mails contenant seulement six caractères (espaces compris !) tels que "1ére ?" ou "2ème ?" et parfois même "3ème ?".
Virginie aussi était dans un autre bureau mais je veux la remercier. Discrète et toujours là, j'espère ne pas l'avoir trop contaminé avec mes dessins de matrices. Je pense aussi à tous les renards qu'elle a abandonné du jour au lendemain...
Merci également à Grégory notre sauveur informatique, même si je l'ai détesté quand j'ai compris qu'il écrivait des programmes en python pour en générer d'autres en C... Merci à lui d'avoir toujours été là pour nous expliquer comment faire alors que l'on pensait la situation désespérée...
Enfin, merci à mes quatre guides touristiques, Irene, Morgan, Nicolas et Rafael. Sans eux mes semaines "ISIT" n'auraient pas été les mêmes. Merci.

Plus personnellement, je souhaite remercier mon frère, Pierre, même s'il est difficile de suivre où il est géographiquement, il est toujours là. Enfin, un immense merci à mes parents pour m'avoir toujours fait confiance et laissé faire ce que je souhaitais.

Sommaire

Notations

Notations génériques

— X^T la transposée de la matrice X
— $(\boldsymbol{u}|\boldsymbol{v})$ la concaténation des vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$
— $\boldsymbol{u}_{i..j} = (u_i, u_{i+1}, \ldots, u_j)$
— $\boldsymbol{u}_{i..j}^{k..l} = \boldsymbol{u}_{i..j}^{k}, \boldsymbol{u}_{i..j}^{k+1}, \ldots, \boldsymbol{u}_{i..j}^{l}$

Notations communes aux deux parties

— $\mathcal{C}$ un code (selon le contexte, il est convolutif ou LDPC)
— n la longueur de $\mathcal{C}$ si c'est un code convolutif
— k la dimension de $\mathcal{C}$ si c'est un code convolutif
— m la mémoire (ou ordre) du code convolutif
— N la longueur de $\mathcal{C}$ si c'est un code LDPC
— K la dimension de $\mathcal{C}$ si c'est un code LDPC
— R le rendement de $\mathcal{C}$
— $\boldsymbol{\mathcal{G}}$ une matrice génératrice
— $\boldsymbol{\mathcal{H}}$ une matrice de parité
— $\rho(x)$ le polynôme définissant la répartition de poids des lignes de la matrice $\boldsymbol{\mathcal{H}}$
— M le nombre de mots bruités observés en sortie de canal
— $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^M$ les M mots d'information
— $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ les M mots de code bruités observés sur le canal
— $\boldsymbol{m}^1, \ldots, \boldsymbol{m}^M$ les M mots de code (non-bruités) transmis sur le canal

Notations de la partie I

— $\boldsymbol{u}$ un mot d'information
— $\boldsymbol{c}$ un mot de code
— Z l'ordre de quasi-cyclicité du code LDPC
— P_i la matrice de taille $Z \times Z$ obtenue en décalant cycliquement la matrice identité de i positions vers la droite
— $\diamond$ une matrice de taille $Z \times Z$ nulle ou obtenue en décalant cycliquement la matrice identité
— $\star$ une matrice de taille $Z \times Z$ cyclique (nulle ou obtenue en sommant une ou plusieurs matrice identité décalées cycliquement)
— (vide) la matrice nulle de taille $Z \times Z$

Notations de la partie II

Notations utilisées dans le chapitre 5

— $\mathcal{C}_1$ et $\mathcal{C}_2$ les deux codeurs convolutifs du turbo-code
— π la permutation interne du turbo-code
— $\tilde{\pi}$ la permutation reconstruite : le plus vraisemblablement celle utilisée
— N la longueur de π
— $\tilde{\boldsymbol{m}}^i = (\tilde{\boldsymbol{u}}^i, \tilde{\boldsymbol{v}}^i, \tilde{\boldsymbol{w}}^i)$ la décomposition de $\tilde{\boldsymbol{m}}^i$ suivant les 3 sorties du turbo-codeur
— p la probabilité d'effacement du canal à effacement
— P la probabilité de reconstruire correctement la permutation
— e_t l'état interne du codeur à l'instant t
— s_t la probabilité de connaître l'état interne du codeur à l'instant t

Notations utilisées dans les chapitres 6 et 7

— t le poids des équations de parité de $\mathcal{C}$
— h une équation de parité de poids t
— c un mot de code
— τ la probabilité d'erreur du canal binaire symétrique
— w le poids des mots recherchés
— π une permutation
— $\mathcal{X}$ la matrice formée des mots bruités (un mot par colonne)
— p, ℓ deux entiers, paramètres de la méthode de recherche d'équations de petits poids
— p_1, p_2 deux entiers (généralement ≤ 3), paramètres de la méthode de Dumer modifiée
— M' un entier, paramètre de la méthode de Dumer modifiée ($0 \leq M' \leq M$)
— N' un entier, paramètre de la méthode de Dumer modifiée ($0 \leq N' \leq N$)

Notations utilisées dans le chapitre 8

— π l'entrelaceur utilisé, il est de longueur N
— $\mathcal{C}_\pi$ l'image de $\mathcal{C}$ par π
— $\mathcal{E} = \{e_1, \ldots, e_t\}$ une équation de parité de poids t
— $\mathcal{E}_{(i)}$ l'équation $\mathcal{E}$ décalée de in positions vers la droite
— $\mathcal{L}$ la liste des équations de parité de poids t retrouvées
— $\mathcal{L}^{NC}$ la liste des équations "non-classées" lors de l'étape de tri
— $\mathcal{L}^1$ la liste des équations du même type utilisées pour la suite de la reconstruction
— t le poids des équations de parité de $\mathcal{C}$
— $\mathcal{P}^{\mathcal{E}}$ le profil de voisinage de l'équation $\mathcal{E}$
— $\mathcal{D}$ un sous code convolutif $(n, n-1)$ du code $\mathcal{C}$
— s_{max} la longueur maximale de l'équation définissant $\mathcal{D}$
— $\tilde{\mathcal{G}}(\mathcal{X})$ le graphe étiqueté associé à l'ensemble d'équations $\mathcal{X}$
— $\tilde{\mathcal{G}}^1(\mathcal{X})$ le sous graphe de $\tilde{\mathcal{G}}(\mathcal{X})$ induit par le voisinage à distance 1 d'un sommet de plus haut degrés
— $\tilde{\mathcal{G}}^2(\mathcal{X})$ le sous graphe de $\tilde{\mathcal{G}}(\mathcal{X})$ induit par le voisinage à distance 2 d'un sommet de plus haut degrés
— ϕ l'isomorphisme défini entre les sommets des graphes
— ψ l'isomorphisme défini entre les étiquettes des graphes

Introduction

Aujourd'hui, les télécommunications numériques sont omniprésentes dans nos vies. Les moyens physiques pour communiquer sont de plus en plus diversifiés avec, par exemple, le téléphone (fixe ou mobile), la radio, la télévision ou encore l'ordinateur ou plus récemment les applications mobiles. Quel que soit le moyen utilisé pour communiquer, les étapes permettant de transmettre un message d'une source à une destination au travers d'un canal de communication se modélisent de la même façon. Ce schéma de transmission est représenté sur la figure 1.

Plus précisément, l'émetteur génère le message qu'il souhaite transmettre, il s'agit d'une suite de symboles binaires. Ce message n'est pas créé aléatoirement, il est donc possible de le compresser afin de réduire la quantité de données à transmettre. Cette étape, appelée codage source, est importante. En effet, la transmission de données a un coût, aussi minime qu'il soit il est indispensable de transmettre la quantité de données minimale.
Le message sera transmis sur un canal de communication. Quel que soit le type de canal utilisé, des erreurs viendront altérer le message transmis. Afin de contrer ce phénomène et que le destinataire soit capable de détecter et, éventuellement, corriger ces erreurs, de la redondance est ajoutée au message compressé. Cette étape, dite de codage canal, est donc elle aussi indispensable et utilise un code détecteur ou correcteur d'erreurs. Ce code doit être adapté au type du canal ainsi qu'à sa probabilité d'erreur. Par exemple, le code utilisé sur un canal où seul 1 bit sur 10 000 est altéré n'est pas le même que pour un canal introduisant dix fois plus de bruit. De plus, la répartition des erreurs est également prise en compte pour le choix du code. En effet, les codes sont plus ou moins performants face à des erreurs se produisant par rafales ou de façon uniforme. Toutes ces caractéristiques sont donc à prendre

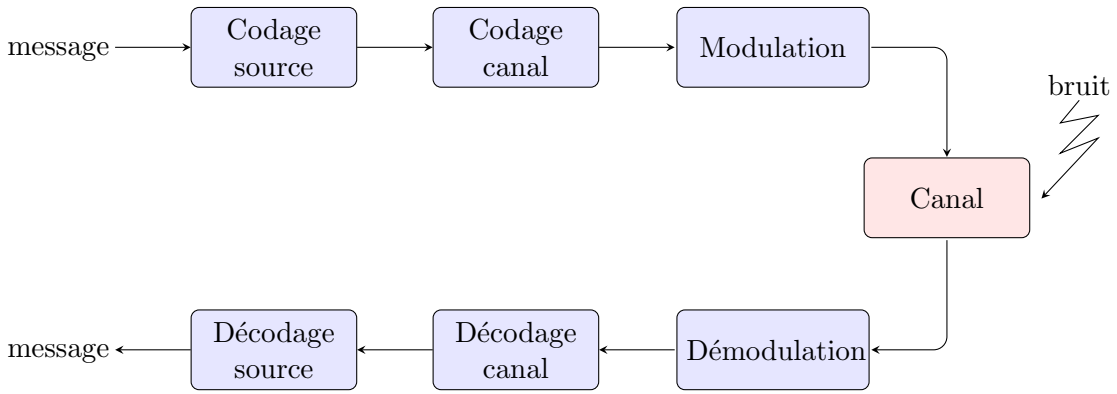


FIGURE 1 – Schéma de transmission numérique

en compte pour choisir le code, c'est également le cas du coût de la transmission et la quantité de données à envoyer. Est-il plus rentable d'ajouter une importante quantité de redondance pour que le destinataire puisse corriger les erreurs, ou ajouter une quantité plus faible, mais retransmettre entièrement le message lorsque celui-ci est trop altéré ? Selon la réponse, on utilisera un code qui est simplement détecteur d'erreurs ou un code correcteur d'erreurs.
La dernière étape avant l'émission sur le canal, est la modulation, celle-ci permet de transformer la suite binaire (compressée et codée) en un signal physique à émettre.

Finalement, le signal physique est transporté sur le canal. Comme expliqué précédemment, il est altéré par du bruit lors de ce transfert. Le message observé en sortie de canal par le destinataire n'est donc pas exactement le même que celui émis. C'est en appliquant les différentes opérations, en sens inverse, que le contenu du message est retrouvé. Il faut donc démoduler, puis décoder (corriger) et enfin décompresser des données reçues.

Afin que le destinataire puisse effectuer ces étapes, il doit connaître les paramètres de chacune d'elles. Autrement dit le destinataire doit connaître le procédé exact de modulation, le code correcteur appliqué et la méthode de compression. En milieu dit coopératif, tous ces paramètres sont effectivement connus par le destinataire. Cependant, ce n'est pas le cas en milieu non-coopératif, il faut alors les retrouver. C'est dans ce contexte que nous nous plaçons : il se produit lorsqu'une communication provenant d'un émetteur inconnu est observée sur un canal. C'est également le cas dans le domaine des télécommunications pour la conception de récepteurs intelligents : pouvoir construire des récepteurs capables de s'adapter automatiquement aux données qu'ils reçoivent est intéressant au sens où il ne serait plus nécessaire de les changer lorsqu'une nouvelle norme de télécommunication est mise en place. Enfin, ce contexte non-coopératif est similaire à celui rencontré en biologie pour la modélisation de la redondance présente dans l'ADN [31, 32, 48, 88, 114].

Reconnaissance du code correcteur

Dans ce manuscrit nous nous intéressons à la reconnaissance du code correcteur, nous supposons donc connus tous les paramètres nécessaires à la démodulation. Les codes utilisés en pratique sont tous des codes linéaires, la détection d'erreurs s'effectuant en temps quadratique par rapport à la longueur des codes utilisés. En 2001, Valembois [131] a montré que la reconnaissance d'un code linéaire de longueur donnée est un problème NP-complet. Une hypothèse supplémentaire est donc nécessaire pour la résolution du problème de la reconnaissance de code, celle-ci porte sur la famille du code utilisé. Par exemple, on supposera que le code à retrouver est un code convolutif, un turbo-code ou encore un code LDPC (low-density parity-check).

Ces dernières années, différents travaux ont été effectués sur la reconnaissance de certaines familles de code. Nous regroupons ci-dessous une liste, quasiment exhaustive, des publications à ce sujet en fonction de l'hypothèse faite sur la famille du code recherché.

— Valembois s'intéresse à la reconnaissance des codes linéaires [130, 131]
— De façon générale, la reconnaissance de codes en bloc est traitée dans les articles [4, 18, 19, 21, 22, 24, 77, 149, 158]

— Des méthodes spécifiques aux codes cycliques sont proposées dans [136, 138, 148, 155]
— La reconnaissance des codes BCH (Bose, Chaudhuri et Hocquenghem) est au centre des articles [82, 135, 137, 139, 154]
— Les codes de Reed-Solomon disposent eux aussi de méthodes de reconnaissance [85, 87, 93, 147, 153]
— Un seul article se consacre à la reconnaissance des codes de Reed-Muller [76]
— La reconnaissance des codes convolutifs fait l'objet de très nombreuses publications [5, 8, 9, 11, 12, 19, 27, 28, 35, 51, 96, 97, 99, 101, 102, 104, 107, 113, 123, 132, 134, 140, 146, 156, 157, 159]
— Celle des codes convolutifs poinçonnés est au centre des articles [20, 23, 94, 95, 103, 109, 124]
— La reconstruction des turbo-codes est étudiée dans [2, 3, 25, 27, 29, 33, 34, 56, 58, 86, 100, 108, 125, 150, 151]
— La reconnaissance des codes LDPC est le sujet des articles [142–145] (dans ces articles, le code est trouvé parmi un ensemble de codes donnés)
— D'autres articles se consacrent à la reconnaissance de l'entrelaceur appliqué après le code correcteur [13, 55, 57, 73–75, 91, 92, 116, 118–121]

Il peut être remarqué l'accroissement récent du nombre de publications sur la reconnaissance de code. Par exemple, parmi les articles cités précédemment, seuls cinq d'entre eux datent d'avant l'an 2000. Et plus de la moitié ont été publiés dans les cinq dernières années. La multiplication des méthodes de reconnaissance montre l'intérêt grandissant pour ce sujet et même si quelques-unes supposent l'utilisation de données non bruitées, ou ne retrouvent qu'une partie des paramètres du code, telle que sa longueur, le problème de la reconnaissance de codes de certaines familles peut être considéré comme résolu. C'est par exemple le cas des codes convolutifs.

Organisation du manuscrit

Ce document est partagé en deux parties, la première porte sur les codes correcteurs et la seconde plus spécifiquement sur la reconnaissance de certaines familles de codes.

Nous commençons donc par rappeler quelques généralités sur les codes linéaires et les différents modèles de canaux, nécessaires à la compréhension des chapitres suivants.
Le chapitre 2 présente de façon un peu plus détaillée les codes convolutifs et les turbo-codes. Ces deux familles faisant partie de celles pour lesquelles nous proposons, par la suite, une méthode de reconnaissance.
Les chapitres 3 et 4 se consacrent aux codes LDPC. Le premier les présente avec leur codage et leur décodage, quant au second il s'intéresse plus particulièrement aux codes LDPC qui sont normés dans les standards de télécommunication. Ces codes sont ainsi regroupés en fonction de leurs structures.

Dans la seconde partie, commençant au chapitre 5, trois méthodes de reconnaissance de codes sont proposées. La première d'entre elles concerne la reconstruction des turbo-codes et plus particulièrement de leur permutation interne. Celle-ci est reconstruite pas à pas en déterminant à chaque étape les valeurs les plus probables des images successives de la permutation

employée. Ce calcul de probabilité s'effectue facilement en utilisant une partie des probabilités calculées dans l'algorithme BCJR [1] de décodage des codes convolutifs. La probabilité de reconstruire correctement la permutation est analysée sur un exemple afin de comprendre le comportement des performances de l'algorithme proposé. Cette méthode est ensuite adaptée et testée pour les turbo-codes poinçonnés puis les turbo-codes série.

Dans le chapitre 6, nous nous intéressons à la reconnaissance des codes LDPC. Pour cela, des équations de parité de petit poids vont être recherchées. Nous présentons rapidement les méthodes existantes sur ce sujet, puis proposons une méthode les améliorant et les généralisant. Celle-ci est basée sur l'algorithme de décodage des codes linéaires de Dumer [39]. La présence d'équations de parité de petit poids caractérise les codes LDPC, mais pas seulement : c'est également le cas des codes convolutifs, même lorsqu'ils sont entrelacés, et donc aussi le cas pour les turbo-codes. La méthode proposée est donc beaucoup plus générale et ne concerne pas uniquement les codes LDPC.
Le chapitre 7 regroupe les résultats de quelques uns des tests effectués sur la reconnaissance des codes LDPC. Les codes reconnus dans ce chapitre correspondent à des codes normés issus du chapitre 4.

Enfin, dans le chapitre 8, nous nous intéressons à la reconnaissance des codes convolutifs entrelacés. Comme dit précédemment, ces codes satisfont de nombreuses équations de parité de petit poids. Ce sont ces équations qui vont être recherchées puis utilisées afin de reconstruire l'entrelaceur. Une fois un ensemble d'équations retrouvé, une nouvelle méthode permettant de reconstruire l'entrelaceur tout en reconnaissant le code convolutif est proposée. Elle est basée sur des considérations de théorie des graphes et ne fait aucune hypothèse sur la structure de l'entrelaceur, qui peut donc être une permutation choisie aléatoirement. Il en est de même pour le code convolutif, aucune hypothèse n'est faite le concernant, il peut même être poinçonné à condition que le motif de poinçonnage soit périodique.

Partie I

Les codes correcteurs d'erreurs

Chapitre 1

Quelques généralités sur les canaux et les codes linéaires

Dans ce chapitre sont regroupées quelques généralités sur les différents modèles de canaux de communication qui seront utilisés par la suite. Dans la seconde section, ce sont quelques définitions et propriétés des codes linéaires qui sont données.

1.1 Canaux de communication

En fonction du canal physique utilisé, différents canaux permettent de modéliser le comportement du bruit altérant les données transportées. Nous présentons, dans cette section, les trois modèles de canaux utilisés afin de tester les performances des méthodes de reconnaissance. Il s'agit des canaux binaires à effacement, symétrique et gaussien.

1.1.1 Canal binaire à effacement de probabilité p

Le canal à effacement de probabilité p peut être illustré par la figure 1.1.

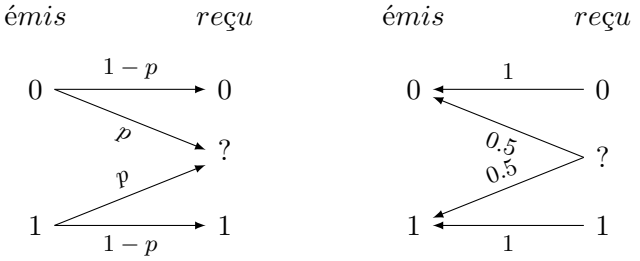


Figure 1.1 – Le canal binaire à effacement de probabilité p.

Sur ce canal, transportant des symboles binaires, un bit est soit transmis correctement soit effacé. En sortie de canal, trois symboles peuvent être observés : les deux valeurs binaires 0 et 1 ainsi que le symbole ? représentant un effacement. L'observation d'un effacement se produit avec une probabilité p, la probabilité qu'un bit soit intact à la réception est donc de $1 - p$.

$$\boldsymbol{P}(reçu = b|émis = b) = 1 - p \text{ et } \boldsymbol{P}(reçu =?|émis = b) = p \text{ pour } b \in \{0, 1\}$$

Lorsque la sortie du canal est un symbole binaire, il est certain que ce symbole n'a pas été altéré durant son transport :

$$\boldsymbol{P}(émis = b|reçu = b) = 1 \text{ pour } b \in \{0, 1\}$$

Par contre, sous l'hypothèse d'une distribution uniforme des bits émis, lorsqu'un effacement est observé en sortie de canal on a aucune information utilisable pour déterminer la valeur du bit émis :

$$\boldsymbol{P}(émis = b|reçu =?) = 1/2 \text{ pour } b \in \{0, 1\}$$

Capacité

La capacité $C_{CE}(p)$ d'un canal à effacement de probabilité p est donnée par (pour plus de détails voir [115], page 12) :

$$C_{CE}(p) = 1 - p$$

1.1.2 Canal binaire symétrique de probabilité d'erreur p

Le canal binaire symétrique de probabilité d'erreur p peut être représenté par la figure 1.2.

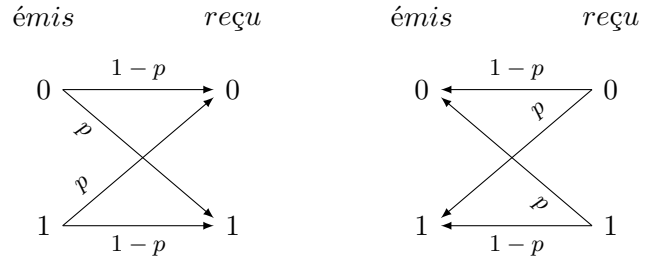


FIGURE 1.2 – Le canal binaire symétrique de probabilité d'erreur p.

Sur ce canal, les symboles émis ainsi que les symboles reçus sont des valeurs binaires. Lorsqu'un symbole est altéré lors de sa transmission, la valeur observée en sortie de canal est le complémentaire de celle émise. Cette inversion de valeur binaire se produit avec une probabilité p. Un bit reste donc intact de l'émission à la réception avec une probabilité égale à $1 - p$.

$$\boldsymbol{P}(reçu = b|émis = b) = 1 - p \text{ et } \boldsymbol{P}(reçu = 1 - b|émis = b) = p \text{ pour } b \in \{0, 1\}$$

Contrairement au canal à effacement, en sortie du canal binaire symétrique il est impossible de savoir de façon certaine la valeur du bit émis connaissant le symbole observé. En effet, ce bit à une probabilité p d'avoir été altéré lors de son transfert :

$$\boldsymbol{P}(émis = 1 - b|reçu = b) = p \text{ pour } b \in \{0, 1\}$$

Par conséquent, la probabilité que la valeur du bit reçu soit bien celle émise est de $1 - p$:

$$\boldsymbol{P}(émis = b|reçu = b) = 1 - p \text{ pour } b \in \{0, 1\}$$

Capacité

La capacité $C_{BS}(p)$ d'un canal binaire symétrique de probabilité d'erreur p est donnée par (pour plus de détails voir [115], page 12) :

$$C_{BS} = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

1.1.3 Canal gaussien de variance σ^2

Sur le canal binaire à bruit blanc gaussien additif, noté BI-AWGN, de variance σ^2, les valeurs observées en sortie de canal sont telles que :

$$reçu = émis + bruit$$

où $émis$ est l'image de la valeur binaire à transmettre par la bijection associant à un symbole binaire une amplitude. En notant a l'amplitude du signal, $émis \in \{-a, a\}$. Quant au bruit, il suit une loi normale centrée en 0 et de variance σ^2 : $bruit \sim \mathcal{N}(0, \sigma^2)$.

La fonction de densité de probabilité du bruit est :

$$p(bruit) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-bruit^2}{2\sigma^2}\right)$$

De cette fonction de densité, il se déduit la probabilité que la valeur émise soit égale à 1 connaissant la valeur reçue :

$$\boldsymbol{P}(émis = 1|reçu) = \frac{1}{1 + \exp(-2a\, reçu/\sigma^2)}$$

Capacité

La capacité C_{CG} d'un canal BI-AWGN de variance σ^2 et d'amplitude $a = 1$ est donnée par (pour plus de détails voir [115], page 14) :

$$C_{CG} = -\int_{-\infty}^{+\infty} p(y)\log_2(p(y))dy - 0.5\log_2(2\pi e\sigma^2)$$

avec

$$p(y) = \frac{1}{2}\left(\boldsymbol{P}(y|x = +1) + \boldsymbol{P}(y|x = -1)\right)$$

où x représente la valeur émise et y la valeur reçue.

Par abus de notations, le canal BI-AWGN est, dans ce document, appelé canal gaussien. Les méthodes utilisées s'appliquent directement aux autres canaux gaussiens, en particulier non binaires.

1.2 Généralités sur les codes linéaires

Les trois familles de code que nous reconstruisons sont des codes linéaires. Nous rappelons quelques généralités sur les codes linéaires avant de détailler plus précisément chacune de ces familles avec leur codage et décodage.

Soit $\mathbb{F}_q$ le corps à q éléments.

Définition 1.1 (Code linéaire)**.** *Un code linéaire $\mathcal{C}$ de longueur n et de dimension k sur $\mathbb{F}_q$ est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Un tel code est noté $[n,k]_q$.*

Tous les codes que l'on considérera par la suite sont définis sur le corps à deux éléments, $\mathbb{F}_2$. Ces codes sont dits binaires et seront simplement notés $[n,k]$.

Définition 1.2 (Rendement)**.** *Le rendement r d'un code $[n,k]_q$ est la proportion d'information contenue dans un mot de code : $r = \frac{k}{n}$.*

En plus de sa longueur et de sa dimension, la capacité de correction d'un code est importante afin d'utiliser un code adapté à la situation considérée.

Définition 1.3 (Distance minimale)**.** *La distance minimale d d'un code $\mathcal{C}$ $[n,k]_q$ est la plus petite distance, au sens de la distance de Hamming, entre deux mots distincts du code $\mathcal{C}$. Pour les codes linéaires, cette distance est égale au plus petit poids des mots non-nuls appartenant au code $\mathcal{C}$.*

Définition 1.4 (Capacité de correction)**.** *Soit $\mathcal{C}$ un code $[n,k]_q$ et d sa distance minimale, la capacité de correction de $\mathcal{C}$ est égale à $\frac{d-1}{2}$. Cette capacité correspond au nombre d'erreurs que le code est capable de corriger dans tous les cas.*

Afin d'associer à tout mot d'information $\boldsymbol{u} = (u_1, \ldots, u_k)$ le mot de code, noté $\boldsymbol{c} = (c_1, \ldots, c_n)$, correspondant, la matrice génératrice du code est utilisée.

Définition 1.5 (Matrice génératrice)**.** *Une matrice génératrice $\boldsymbol{\mathcal{G}}$ d'un code linéaire $\mathcal{C}$ $[n,k]_q$ est une matrice de taille $k \times n$ à coefficients dans $\mathbb{F}_q$. Les lignes de cette matrice forment une base de $\mathcal{C}$.*

Une méthode de codage des codes linéaires consiste à effectuer le produit $\boldsymbol{u\mathcal{G}}$. Quant au décodage, il peut utiliser la matrice de parité :

Définition 1.6 (Matrice de parité)**.** *La matrice de parité $\boldsymbol{\mathcal{H}}$ d'un code linéaire $\mathcal{C}$ $[n,k]_q$ est une matrice de n colonnes et de rang $n-k$ à coefficients dans $\mathbb{F}_q$ telle que pour tout mot de code $\boldsymbol{c} \in \mathcal{C}$, $\boldsymbol{\mathcal{H}c}^T = \boldsymbol{0}$.*
Le code engendré par la matrice $\boldsymbol{\mathcal{H}}$ est un code $[n, n-k]_q$ appelé code dual de $\mathcal{C}$, il est noté $\mathcal{C}^\perp$.

Par définition, une matrice de parité $\boldsymbol{\mathcal{H}}$ d'un code linéaire $\mathcal{C}$ engendré par la matrice génératrice $\boldsymbol{\mathcal{G}}$ est telle que $\boldsymbol{\mathcal{G}\mathcal{H}}^T = \boldsymbol{0}$. La forme de la matrice $\boldsymbol{\mathcal{G}}$ peut être utilisée pour déterminer $\boldsymbol{\mathcal{H}}$.

Définition 1.7 (Matrice génératrice systématique)**.** *La matrice génératrice $\boldsymbol{\mathcal{G}}$ est dite systématique si elle est de la forme $(I_k|A)$, où I_k représente la matrice identité de taille $k \times k$. Un codage effectué par une telle matrice est dit systématique.*

Dans un codage systématique, le mot de code associé au mot d'information $\boldsymbol{u}$ contient $\boldsymbol{u}$ sur ses k premiers indices. La matrice de parité duale se déduit d'une matrice génératrice systématique :

Proposition 1.8. *Soit $\mathcal{C}$ un code linéaire engendré par la matrice génératrice systématique $\boldsymbol{\mathcal{G}} = (I_k|A)$. La matrice $\boldsymbol{\mathcal{H}}$ définie par $(-A^T|I_{n-k})$ est une matrice de parité du code $\mathcal{C}$.*

Chapitre 2

Les codes convolutifs et les turbo-codes

Les codes convolutifs sont une grande famille de codes que l'on oppose généralement aux codes en bloc. La différence entre ces deux types de codes porte sur les relations entre l'information à coder et les mots de code générés. Pour les codes en bloc, le codage d'un mot d'information n'est pas impacté par le codage des mots d'information précédents. Les codes de Hamming, de Reed-Solomon ou encore les codes LDPC sont des codes en bloc, c'est également le cas des turbo-codes.

Cette structure par blocs n'est donc pas vérifiée par les codes convolutifs. En effet, le codage du i-ème mot d'information dépend du ou des mots d'information précédents. Ce principe se visualise grâce au codeur utilisant des registres à décalage. Lorsque de l'information entre dans le codeur, l'information précédemment entrée est encore stockée dans les registres et elle est utilisée afin de déterminer le mot de code. En opposition au codage en bloc, ce type de codage peut être qualifié de codage par "flux".

2.1 De la théorie de Shannon aux codes convolutifs

Les codes convolutifs sont toujours très largement utilisés en pratique. Ils ont même pendant longtemps dominés les codes en bloc suite à une longue évolution de la théorie de l'information introduite par Shannon en 1948 [117] comme l'a expliqué Forney dans sa "Shannon-Lecture" de 1995 [52].

Effectivement, Shannon a posé les problèmes fondamentaux de la théorie de l'information dans [117], et a, dans ce même article, répondu à ces problèmes mais de façon large et théorique. Les travaux futurs en théorie de l'information partent du résultat principal de Shannon : pour tout canal sans mémoire de capacité C, il existe un code de rendement $R < C$ ainsi qu'un algorithme de décodage associé à ce code tel que la probabilité d'erreur après décodage $\boldsymbol{P}(E)$ soit aussi faible que souhaitée. Il existe donc de "bons" codes, mais dans sa preuve Shannon ne construit pas de tels codes et laisse donc ouvert le problème de leur construction.

Dans cette même preuve, Shannon suppose l'utilisation d'un décodage exhaustif au maximum de vraisemblance. Le problème de ce type de décodage est sa complexité qui est proportionnelle au nombre de mots contenus dans le code considéré. Or pour approcher la capacité du canal il est nécessaire d'utiliser des codes longs, ce type de décodage n'est alors pas utilisable en pratique. D'où la nécessité de développer également de nouvelles méthodes de décodage.

À l'époque, les recherches se sont alors portées sur deux principaux problèmes : comment $\boldsymbol{P}(E)$ tend vers 0, et comment approcher, en pratique, la capacité du canal. Pour le premier de ces problèmes, Gallager a apporté en 1965 [54] une réponse concernant les codes en bloc : la probabilité d'erreur après décodage du meilleur code en bloc de longueur N et de rendement R décroît exponentiellement avec N : $\boldsymbol{P}(E) \cong e^{-NE(R)}$ où $E(R)$ est positif pour tout $R < C$. Tout comme Shannon, Gallager suppose l'utilisation d'un décodage exhaustif au maximum de vraisemblance, la complexité G de ce décodage est de l'ordre de $G \cong e^{NR}$. Il s'en déduit que $\boldsymbol{P}(E)$ décroît seulement algébriquement par rapport à la complexité du décodage pour les codes en bloc : $\boldsymbol{P}(E) \cong G^{-E(R)/R}$.
Concernant le second problème, la difficulté principale vient de la complexité du décodage, et non de la construction de "bons" codes longs, même si cette construction n'est pas évidente. Pour résoudre ce problème il a alors été proposé d'utiliser un long code convolutif choisi aléatoirement avec un décodage séquentiel. Les codes convolutifs ont une structure dynamique, qui, à cette époque, c'est-à-dire au début des années 60, était représentée sous forme d'un arbre. Le décodage séquentiel utilise cette structure très régulière des codes convolutifs. En effet, si un code convolutif est entièrement poinçonné sauf sur N' positions consécutives les mots de code obtenus satisfont tout de même un ensemble d'équations de parité. Cette structure régulière permet d'élaguer dans l'arbre des possibilités en considérant uniquement des petits codes poinçonnés. Finalement le nombre de chemins conservés reste raisonnable jusqu'à la fin du décodage. Avec un choix de paramètres optimaux, la probabilité que le nombre de calculs nécessaires au décodage dépasse un certain nombre G suit une distribution algébrique de Pareto : $\boldsymbol{P}(calcul > G) \cong G^{-\alpha(R)}$. L'exposant de Pareto $\alpha(R)$ est supérieur à 1 et si $R < R_0$ alors le nombre moyen de calculs est borné ; R_0 est le seuil pour lequel le décodage peut être effectué de façon pratique. En fait, le décodage séquentiel est une méthode efficace pour laquelle il a été prouvée qu'elle atteint une erreur de probabilité nulle sur tous les canaux sans mémoire et pour tout code de rendement $R < R_0$.

Ce seuil R_0, appelé en anglais "cutoff rate", a alors été considéré comme étant la "capacité pratique" du canal. L'utilisation d'un décodage séquentiel sur un code convolutif suffisamment long et de rendement $R < R_0$ permettant d'atteindre plus ou moins cette "capacité pratique" le problème posé par Shannon a alors été considéré comme résolu.

Pour un code convolutif suffisamment long les erreurs de décodage se produisent lorsque le nombre de calculs dépasse une certaine limite de complexité G. Ainsi, la probabilité d'erreur après décodage des codes convolutifs décroît algébriquement par rapport à G. C'est également le cas pour les codes en bloc, mais la complexité de ce décodage étant inférieure, les codes convolutifs sont alors préférés aux codes en bloc.

Un autre problème théorique était de déterminer le comportement de la probabilité d'erreur des codes convolutifs de façon similaire au résultat de Gallager sur les codes en bloc. C'est Viterbi [133] et Yudkin [152] qui ont introduit le résultat suivant : $\boldsymbol{P}(E) \cong e^{-\nu e(R)}$

où ν est la longueur de contrainte du code convolutif et l'exposant $e(R)$ est supérieur à 0 pour tout $R < C$. Plus précisément, $e(R) = R_0$ pour tout rendement $R < R_0$ et $e(R)$ dépasse l'exposant $E(R)$ des codes en bloc pour $0 < R < C$, en effet, $lim_{R \to C} e(R)/E(R) = +\infty$.

L'algorithme de Viterbi, qui est de nos jours très largement utilisé, est apparu pour la première fois comme preuve technique du résultat théorique précédent. En utilisant cet algorithme, qui a ensuite été présenté comme une technique de décodage au maximum de vraisemblance à partir d'un treillis, la complexité G du décodage est : $G \cong e^{\nu R}$. D'où une probabilité d'erreur $\boldsymbol{P}(E) \cong G^{-e(R)/R}$, et bien que l'exposant soit inférieur à celui des codes en bloc, le comportement de $\boldsymbol{P}(E)$ est lui toujours algébrique par rapport à la complexité. Il est intéressant de remarquer que cet exposant est égal à l'exposant de Pareto apparaissant pour le décodage séquentiel, le compromis performance/complexité du décodage séquentiel est donc asymptotiquement égal à celui de l'algorithme de Viterbi.

Le fait que $e(R) > E(R)$ a été utilisé pour mettre en avant que les codes convolutifs sont intrinsèquement supérieurs aux codes en bloc. En effet, la relation entre $e(R)$ et $E(R)$ est liée au fait que tout code en bloc optimal de rendement $R < C$ peut être construit en terminant proprement un code convolutif aléatoire. Cependant, ceci implique également que si un code en bloc est vu comme un code convolutif terminé alors il peut être décodé grâce à l'algorithme de Viterbi et donc avec une complexité : $G \cong e^{\nu R}$ et non en $G \cong e^{NR}$ comme prévu pour les codes en bloc. Par conséquence, la seule différence sur le compromis performance/complexité entre un code convolutif et un code en bloc vu comme un code convolutif terminé est le rendement perdu par la phase de terminaison pour les codes en bloc. Mais cette perte pouvant être réduite autant que souhaitée en utilisant de longs codes, il n'y a asymptotiquement pas de différence pratique entre un code convolutif et un code en bloc sur le compromis performance/complexité.

Ce n'est qu'en 1993 et grâce à la découverte des turbo-codes que la domination des codes convolutifs va diminuer. En effet, les turbo-codes inventés par Berrou et Glavieux [10] avec leur décodage itératif atteignent une probabilité d'erreur très faible pour des rendements bien au delà de R_0 (la borne considérée jusqu'ici comme la "capacité pratique"). La capacité de ces codes vient du fait que l'utilisation d'une permutation interne permet à la probabilité d'erreur de ne plus être bornée en fonction de la longueur de contrainte ν du code convolutif utilisé [6] contrairement aux codes convolutifs pour lesquels $\boldsymbol{P}(E) \cong e^{-\nu e(R)}$. Finalement, la permutation interne permet aux turbo-codes d'avoir d'excellentes performances et d'approcher de très près la capacité du canal tout en ayant un décodage très efficace.

2.2 Codage des codes convolutifs

Nous nous intéressons uniquement au cas binaire, un code convolutif $\mathcal{C}$ de longueur n et de dimension k possède une matrice génératrice de k lignes et n colonnes, dont chaque coefficient est un polynôme de $\mathbb{F}_2[D]$. Cette matrice génératrice permet de définir le code convolutif mais ne permet pas l'encodage.

Le codage peut s'effectuer de plusieurs façons, la plus courante étant celle utilisant des registres à décalage, mais il peut également être réalisé en effectuant le produit de l'information par une matrice génératrice binaire infinie ou en utilisant un treillis de codage (autrement dit

un graphe).

Un codeur convolutif binaire de longueur n, de dimension k, noté (n, k) et d'ordre m est défini par $m+1$ matrices binaires $\boldsymbol{\mathcal{G}}_0, \ldots, \boldsymbol{\mathcal{G}}_m$ de taille $k \times n$. Le code convolutif $\mathcal{C}$ est l'ensemble des mots obtenus en effectuant le produit de l'information $\boldsymbol{u}$ par la matrice binaire infinie $\boldsymbol{\mathcal{G}}$:

$$\boldsymbol{\mathcal{G}} = \begin{pmatrix} \boldsymbol{\mathcal{G}}_0 & \boldsymbol{\mathcal{G}}_1 & \cdots & \boldsymbol{\mathcal{G}}_m & & & & \\ & \boldsymbol{\mathcal{G}}_0 & \boldsymbol{\mathcal{G}}_1 & \cdots & \boldsymbol{\mathcal{G}}_m & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & \boldsymbol{\mathcal{G}}_0 & \boldsymbol{\mathcal{G}}_1 & \cdots & \boldsymbol{\mathcal{G}}_m & \\ & & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

Autrement dit, le mot de code $\boldsymbol{c} = (c_0\, c_1 \ldots)$ est défini par $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{\mathcal{G}} = (u_0\, u_1 \ldots)\boldsymbol{\mathcal{G}}$ où $\forall i, u_i \in \mathbb{F}_2^k$ et $c_i \in \mathbb{F}_2^n$. D'où :

$$\begin{aligned} c_0 &= u_0\boldsymbol{\mathcal{G}}_0 \\ c_1 &= u_0\boldsymbol{\mathcal{G}}_1 + u_1\boldsymbol{\mathcal{G}}_0 \\ c_2 &= u_0\boldsymbol{\mathcal{G}}_2 + u_1\boldsymbol{\mathcal{G}}_1 + u_2\boldsymbol{\mathcal{G}}_0 \\ &\vdots \\ c_{i+m} &= \sum_{j=0}^{i} u_{i+j}\boldsymbol{\mathcal{G}}_{m-j} \end{aligned}$$

c_i (le i-ème bloc de n bits du mot de code) dépend donc des m blocs de k bits d'information précédents : $u_{i-m}, u_{i-m+1}, \ldots, u_i$.

Par construction, les codes convolutifs vérifient de nombreuses équations de parité de petit poids. En effet, la matrice de parité $\boldsymbol{\mathcal{H}}$ d'un code convolutif est une matrice binaire infinie possédant la même structure que la matrice génératrice :

$$\boldsymbol{\mathcal{H}} = \begin{pmatrix} \boldsymbol{\mathcal{H}}_0 & \boldsymbol{\mathcal{H}}_1 & \ldots & \boldsymbol{\mathcal{H}}_m & & & & \\ & \boldsymbol{\mathcal{H}}_0 & \boldsymbol{\mathcal{H}}_1 & \ldots & \boldsymbol{\mathcal{H}}_m & & & \\ & & \ddots & \ddots & & \ddots & & \\ & & & \boldsymbol{\mathcal{H}}_0 & \boldsymbol{\mathcal{H}}_1 & \ldots & \boldsymbol{\mathcal{H}}_m & \\ & & & & \ddots & \ddots & & \ddots \end{pmatrix}$$

où les sous-matrices $\boldsymbol{\mathcal{H}}_i$ sont de taille $(n-k) \times n$. Les mots de code satisfont donc $n-k$ équations de parité qui, décalées d'un multiple de n vers la droite sont encore vérifiées.

Exemple. *Le même exemple est suivi tout au long de cette section. Soit le code convolutif* $(2, 1)$ *défini par la matrice génératrice* $\boldsymbol{\mathcal{G}}$ *:*

$$\boldsymbol{\mathcal{G}} = \begin{pmatrix} 11 & 11 & 01 & & & \\ & 11 & 11 & 01 & & \\ & & 11 & 11 & 01 & \\ & & & 11 & 11 & 01 \\ & & & & \ldots & \end{pmatrix}$$

La suite binaire codée $\boldsymbol{c}$ associée à la suite d'information $\boldsymbol{u} = (0101100\ldots0)$ peut se calculer en effectuant le produit matriciel $\boldsymbol{u}\boldsymbol{\mathcal{G}}$. D'où $\boldsymbol{c} = (0011111000100100\ldots0)$. Les suites $\boldsymbol{u}$ et $\boldsymbol{c}$ sont infinies, c'est pour cette raison qu'elles sont généralement tronquées, si $\boldsymbol{u}$ est de longueur "significative" égale à αk alors les αn premiers bits de $\boldsymbol{c}$ sont conservés. Sur cet exemple, nous supposons que l'information est contenue sur les 6 premiers bits de $\boldsymbol{u}$, la suite de bits transmise est alors 001111100010.

Un codeur convolutif peut également être représenté par la matrice polynomiale $\boldsymbol{\mathcal{G}}(D)$ de taille $k \times n$ définie par :

$$\begin{aligned}\boldsymbol{\mathcal{G}}(D) &= \sum_{i=0}^{m} \boldsymbol{\mathcal{G}}_i D^i \\ &= \begin{pmatrix} g_{1,1}(D) & g_{1,2}(D) & \cdots & g_{1,n}(D) \\ g_{2,1}(D) & g_{2,2}(D) & \cdots & g_{2,n}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,1}(D) & g_{k,2}(D) & \cdots & g_{k,n}(D) \end{pmatrix}\end{aligned}$$

L'ordre m du codeur est alors égal à $m = \max_{i,j}(\deg g_{i,j})$.

Exemple. *La matrice génératrice $\boldsymbol{\mathcal{G}}(D)$ associée à la matrice génératrice binaire infinie $\boldsymbol{\mathcal{G}}$ de l'exemple ci-dessus est : $\boldsymbol{\mathcal{G}}(D) = (1 + D \quad 1 + D + D^2)$.*

D'autres méthodes existent pour déterminer la suite codée correspondant à une suite de bits d'information. La première consiste à utiliser un codeur à registres à décalage. Ces registres sont tous initialisés à 0 au début du codage. À chaque top d'horloge, k nouveaux bits entrent dans le codeur et n bits de sortie sont déterminés en fonction de l'entrée et des valeurs stockées dans les registres. Les contenus des registres sont ensuite décalés d'un cran, la valeur la plus ancienne sort du codeur et n'influera pas sur les prochaines sorties.

Définition 2.1 (Mémoire d'un codeur)**.** *La mémoire d'un code convolutif est le nombre minimal de registres à décalage nécessaires pour construire le codeur correspondant.*

Exemple. *Toujours sur le même exemple, un codeur à registres à décalage permettant de générer le code de matrice génératrice $\boldsymbol{\mathcal{G}}$ est représenté sur la figure 2.1. Afin de coder le message 010110, le bit 0 est entré dans le codeur, les deux premiers bits de sortie sont déterminés : 00 (ils dépendent du 0 entré ainsi que du contenu des registres, qui sont initialisés à 0). Le contenu des registres est décalé, l'état interne est encore égal à 00. Un nouveau bit d'information est alors entré dans le codeur, il s'agit d'un 1, la sortie correspondante est 11. Les registres sont mis à jour, l'état interne du codeur est à cet instant 10. Et ainsi de suite jusqu'à avoir entré les 6 bits d'information, la sortie générée est alors 00 11 11 10 00 10.*

Une autre méthode de codage est basée sur le diagramme d'états du codeur à registres à décalage. Dans un diagramme d'états, tous les états internes possibles du codeur sont représentés par un sommet. S'il est possible de passer d'un état à un autre en un seul top d'horloge, une arête relie ces deux états. De plus, les arêtes sont étiquetées par l'entrée permettant d'effectuer ce changement d'état ainsi que par la sortie générée. Le codage consiste alors à suivre un chemin dans ce graphe, en partant de l'état interne initial égal à $00\ldots0$.

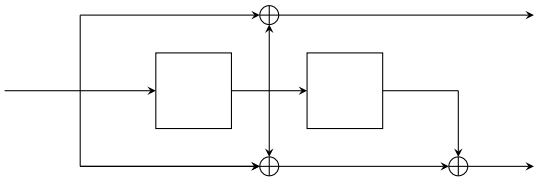


FIGURE 2.1 – Codeur à registres à décalage de matrice génératrice
$\boldsymbol{\mathcal{G}}(D) = (1 + D \quad 1 + D + D^2)$

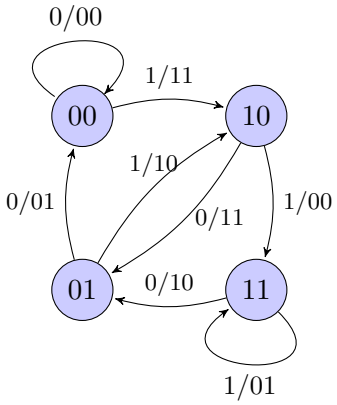


FIGURE 2.2 – Diagramme d'états correspondant au codeur de la figure 2.1

Exemple. *La figure 2.2 représente le diagramme d'états associé au codeur à registres à décalage de la figure 2.1. Sur ce diagramme, la notation a/bc indique qu'il faut entrer a dans le codeur pour effectuer le changement d'état correspondant et que la sortie générée est bc. Le codage de* 010110 *commence à l'état* 00 *puisque les registres sont tous initialisés à* 0*. Le premier bit à entrer est* 0*, l'état du registre ne varie donc pas et la sortie émise à cet instant est* 00*. Le bit suivant est un* 1*, le codeur passe donc à l'état* 10 *et émet* 11*, ... Finalement, la suite codée obtenue est* 00 11 11 10 00 10*.*

Enfin, la dernière façon de coder consiste à utiliser un treillis. Un treillis correspond à la concaténation d'une infinité de diagrammes d'états dépliés. Tout chemin allant de gauche à droite dans ce treillis correspond à un mot de code. Le codage s'effectue donc en suivant un chemin le long de ce treillis.

Exemple. *Le treillis de codage correspondant à l'exemple suivi est représenté sur la figure 2.3. Le codage de la suite binaire* 010110 *suit donc le chemin mis en évidence sur la figure 2.4. En lisant sur chaque arête la sortie correspondante, la suite codée obtenue est* 00 11 11 10 00 10*.*

2.3 Décodage des codes convolutifs

Deux algorithmes sont principalement utilisés pour décoder les codes convolutifs ; l'algorithme de Viterbi [133] et l'algorithme BCJR [1]. Ce dernier sert essentiellement dans des algorithmes de décodage itératif.

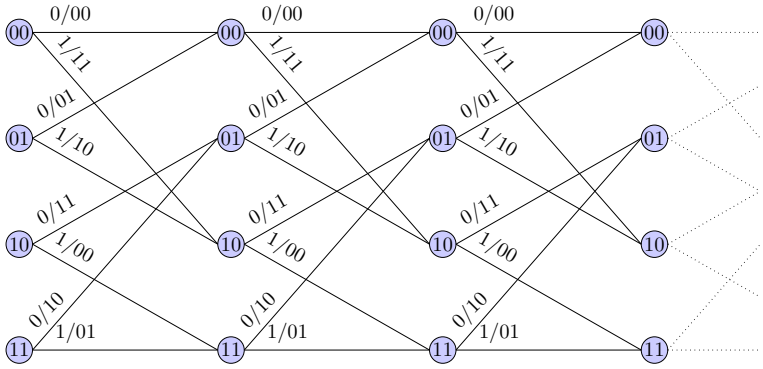


FIGURE 2.3 – Treillis de codage correspondant au codeur de la figure 2.1

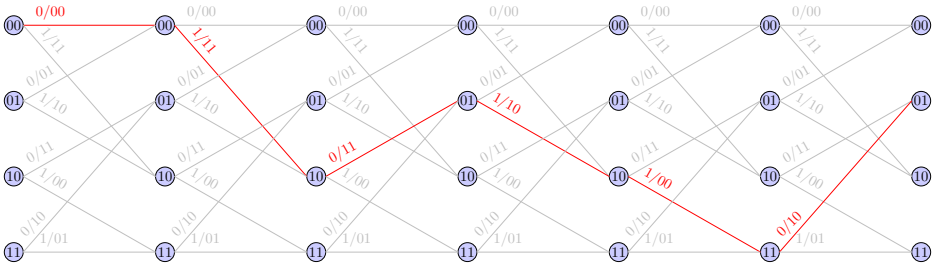


FIGURE 2.4 – Chemin de codage de 010110

2.3.1 Algorithme de Viterbi

Le principal algorithme de décodage des codes convolutifs est celui de Viterbi [133], il recherche le mot de code le plus probable connaissant le mot reçu en sortie d'un canal sans mémoire. Autrement dit, il recherche le mot de code à la plus petite distance de Hamming du mot reçu. Pour cela, les arêtes du treillis du code convolutif sont pondérées par leur distance avec le mot reçu, le décodage consiste alors à rechercher dans ce graphe pondéré le plus court chemin allant de gauche à droite. Cette recherche du plus court chemin est un problème classique de théorie des graphes dont la résolution possède une complexité polynomiale.

Exemple. *Soit le mot reçu* 00 11 01 10 00 10*. En reprenant le codeur convolutif des exemples de la section précédente, le treillis pondéré obtenu est représenté sur la figure 2.5. Chaque indice noté en bleu indique donc la distance entre les bits émis en suivant cette arête et les bits reçus correspondants. Sur cette même figure, le chemin le plus court est celui mis en rouge. De ce chemin, il se déduit le mot de code le plus probablement émis :* 00 11 11 10 00 10 *qui est seulement à distance* 1 *du mot reçu.*

2.3.2 Algorithme de BCJR

L'algorithme dit BCJR [1], proposé en 1974, doit son nom à ses auteurs : Bahl, Cocke, Jenlinek et Raviv. Cet algorithme est principalement utilisé dans les méthodes de décodage itératif des turbo-codes ainsi que pour le décodage de certains codes LDPC structurés.

BCJR recherche l'information la plus probablement codée et non le mot de code le plus probable comme dans l'algorithme de Viterbi. L'objectif est donc de déterminer pour tout $t \in \{1, \ldots, N\}$ la probabilité $\boldsymbol{P}(u_t = i | \text{les données reçues})$ (avec $i \in \{0, 1\}$), pour cela deux

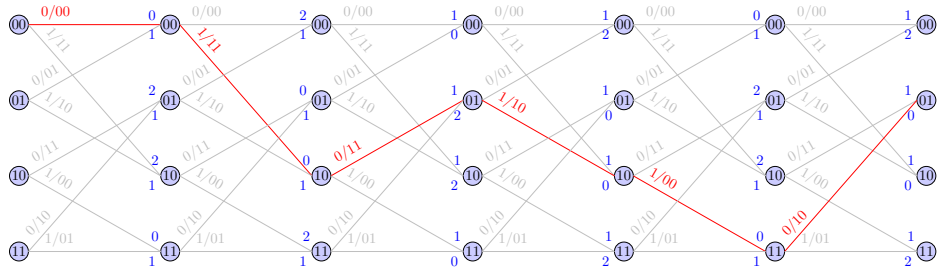


FIGURE 2.5 – Décodage de Viterbi du mot reçu 00 11 01 10 00 10

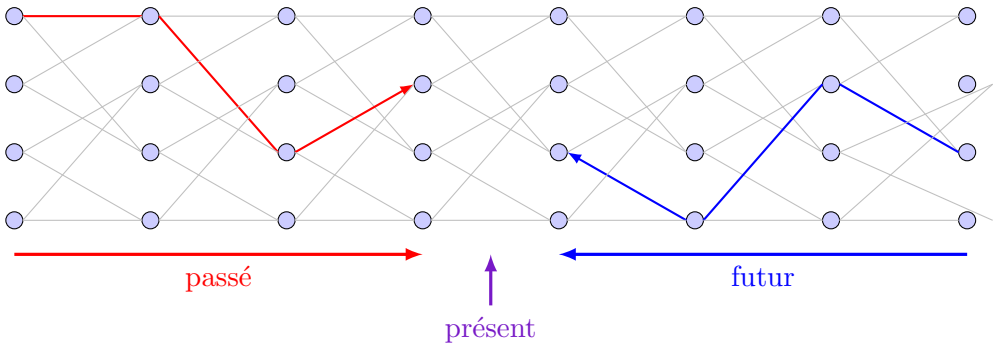


FIGURE 2.6 – Schéma de l'algorithme de décodage BCJR

passes sont effectuées le long du treillis, l'une de gauche à droite et l'autre en sens inverse. Le fait que cette méthode retourne des probabilités sur chacun des bits d'information permet d'utiliser directement sa sortie comme une entrée (sous forme d'information souple) d'un autre décodeur. Cela explique son utilisation dans des décodages itératifs.

Pour déterminer la valeur la plus probable du bit d'information u_t entré dans le codeur à l'instant t, l'algorithme de BCJR se décompose en plusieurs étapes. La première, calcule les probabilités de tous les changements d'état entre les instants $t-1$ et t. Lors de ces calculs seules les probabilités des bits émis lors de ce changement d'état sont considérées. On dit que cette étape prend en compte le présent.
La seconde étape détermine la probabilité de chacun des états interne du codeur à l'instant $t-1$ en fonction des probabilités des bits précédemment reçus (c'est-à-dire de tous les bits reçus entre l'instant 0 et l'instant $t-1$). Cette étape prend donc en compte le passé.
La troisième, calcule elle aussi les probabilités de chaque état interne du codeur mais à l'instant t et tient compte uniquement des bits reçus ultérieurement (de l'instant $t+1$ jusqu'à la fin). C'est pour cette raison que l'on dit que cette étape utilise le futur.
Enfin, la dernière étape de l'algorithme fait une synthèse des différentes probabilités calculées lors des étapes précédentes afin de déterminer la valeur la plus probable de u_t.

Ces étapes sont maintenant décrites plus précisément et en particulier avec les calculs des probabilités. Afin de simplifier les écritures et les représentations, considérons le cas d'un code convolutif $(2,1)$. L'algorithme BCJR s'appliquant parfaitement aux codes convolutifs de dimension et/ou longueur supérieures. De plus, les mots d'information sont supposés aléatoires et uniformément répartis. Cette méthode correspond à l'algorithme 1.

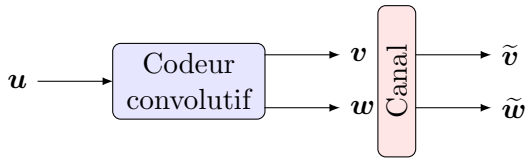


FIGURE 2.7 – Notations des entrées et sorties du codeur convolutif et du canal

Notations. La figure 2.7 regroupe les notations utilisées dans le suite de cette sous-section. Tous les vecteurs d'entrées et sorties sont de même taille, notée N, leurs indices sont numérotés de 1 à N. Soient m la mémoire du code convolutif et $\widetilde{\boldsymbol{u}}$ l'information la plus vraisemblablement émise connaissant $\widetilde{\boldsymbol{v}}$ et $\widetilde{\boldsymbol{w}}$: $\widetilde{\boldsymbol{u}}$ est la sortie de l'algorithme BCJR. L'état interne du codeur à l'instant t est noté e_t.
Enfin, la notation $\widetilde{\boldsymbol{v}}_{i..j}$ représente le vecteur $(\widetilde{v}_i, \ldots, \widetilde{v}_j)$, de même pour $\widetilde{\boldsymbol{w}}_{i..j}$.

A chaque étape de l'algorithme les probabilités pour tous les instants sont déterminées simultanément afin d'éviter d'effectuer de nombreuses fois les mêmes calculs. Toutes ces valeurs sont stockées dans des tableaux.

Première étape : en fonction du présent

Lors de cette étape, le tableau noté γ est complété. Il s'agit d'un tableau de N colonnes, représentant les différents instants, et 2×2^m lignes, chacune correspondant à un changement d'état interne du codeur.

Le but de cette étape est donc de déterminer, pour tous les instants $t \in \{1, \ldots, N\}$ et tous les couples (a, b) tels que le codeur peut passer de l'état a à l'état b, la probabilité :

$$\gamma_{a,b}^t = \boldsymbol{P}(e_t = b | e_{t-1} = a, \widetilde{v_t} \text{ et } \widetilde{w_t})$$

En notant ij la sortie émise par le codeur lors de son passage de l'état a à l'état b, il se déduit :

$$\gamma_{a,b}^t = \boldsymbol{P}(v_t = i \text{ et } w_t = j | \widetilde{v_t} \text{ et } \widetilde{w_t})$$

De l'utilisation d'un canal sans mémoire il découle :

$$\gamma_{a,b}^t = \boldsymbol{P}(v_t = i | \widetilde{v_t}) \boldsymbol{P}(w_t = j | \widetilde{w_t})$$

Les probabilités $\boldsymbol{P}(v_t = i | \widetilde{v_t})$ et $\boldsymbol{P}(w_t = j | \widetilde{w_t})$ se déduisent simplement des données observées $\widetilde{v_t}$ et $\widetilde{w_t}$. Leur calcul dépend du type de canal utilisé, comme expliqué dans le chapitre 1.

Deuxième étape : en fonction du passé

Durant cette étape, le tableau noté α est complété, il contient $N+1$ colonnes et 2^m lignes. Le coefficient placé ligne a colonne t contient la probabilité :

$$\alpha_a^t = \boldsymbol{P}(e_t = a | \widetilde{\boldsymbol{v}}_{1..t} \text{ et } \widetilde{\boldsymbol{w}}_{1..t})$$

Il est obtenu via la formule de récurrence :

$$\alpha_a^t = \eta \sum_{b=0}^{2^m-1} \boldsymbol{P}(e_{t-1} = b | \widetilde{\boldsymbol{v}}_{1..t-1} \text{ et } \widetilde{\boldsymbol{w}}_{1..t-1}) \boldsymbol{P}(e_t = a | e_{t-1} = b, \widetilde{v_t} \text{ et } \widetilde{w_t})$$

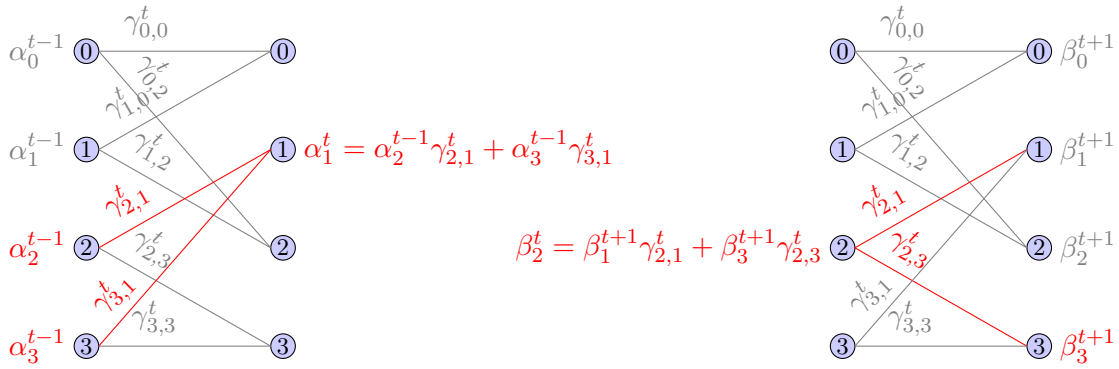


FIGURE 2.8 – Illustration des calculs de α^t et β^t

où η est un coefficient de normalisation tel que :

$$\sum_{a=0}^{2^m-1} \alpha_a^t = 1$$

En utilisant les tableaux α et γ, on obtient :

$$\alpha_a^t = \eta \sum_{b=0}^{2^m-1} \alpha_b^{t-1}\gamma_{b,a}^t$$

La situation est représentée sur la figure 2.8. Cette probabilité concernant l'instant t dépend des probabilités liées à l'instant précédent. Il est donc important de déterminer les valeurs des α_a^0 pour tous les états internes a. Si l'initialisation des registres du codeur est connue, la probabilité de l'état correspondant est de 1 et celles des autres états sont nulles. Si l'initialisation du codeur n'est a priori pas connue, tous les états sont équiprobables : $\alpha_a^0 = \frac{1}{2^m}$, $\forall a$.

Troisième étape : en fonction du futur

Cette dernière étape intermédiaire est très proche de la précédente. Le tableau correspondant est noté β, il est composé de $N+1$ colonnes et de 2^m lignes. L'élément, noté β_a^t, à l'intersection de la ligne a et de la colonne t est égal à la probabilité :

$$\beta_a^t = \boldsymbol{P}(e_t = a | \widetilde{\boldsymbol{v}}_{t+1..N} \text{ et } \widetilde{\boldsymbol{w}}_{t+1..N})$$

Comme pour l'étape dépendant du passé, cette probabilité se décompose en une somme :

$$\begin{aligned} \beta_a^t &= \lambda \sum_{b=0}^{2^m-1} \boldsymbol{P}(e_{t+1} = b | \widetilde{\boldsymbol{v}}_{t+2..N} \text{ et } \widetilde{\boldsymbol{w}}_{t+2..N}) \boldsymbol{P}(e_t = a | e_{t+1} = b, \widetilde{v}_{t+1} \text{ et } \widetilde{w}_{t+1}) \\ &= \lambda \sum_{b=0}^{2^m-1} \beta_b^{t+1}\gamma_{a,b}^{t+1} \end{aligned}$$

où λ est une constante de normalisation telle que :

$$\sum_{a=0}^{2^m-1} \beta_a = 1$$

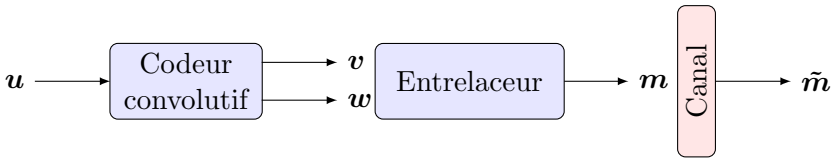


FIGURE 2.9 – Entrelacement des codes convolutifs

La figure 2.8 schématise également le calcul de β^t. Pour déterminer β_a^t il est nécessaire de connaître les valeurs des β^{t+1}. L'initialisation se fait donc en fonction de l'état interne final du codeur. Si celui-ci est connu un seul β^N est égal à 1, les autres sont nuls, c'est par exemple le cas pour un codage en tail-biting. Si le dernier état interne du codeur n'est pas connu, tous les états sont équiprobables.

Quatrième étape : synthèse

L'objectif est de déterminer, pour chaque t, la probabilité $\boldsymbol{P}(u_t = 0|\widetilde{\boldsymbol{v}} \text{ et } \widetilde{\boldsymbol{w}})$. Si cette valeur est supérieure à 1/2 alors $\widetilde{u}_t = 0$ sinon $\widetilde{u}_t = 1$. Les calculs intermédiaires précédents impliquent :

$$\boldsymbol{P}(u_t = i|\widetilde{\boldsymbol{v}} \text{ et } \widetilde{\boldsymbol{w}}) = \rho \sum_{(a,b):a\xrightarrow{i}b} \alpha_a^{t-1}\gamma_{a,b}^t\beta_b^t$$

où ρ est une constante telle que : $\boldsymbol{P}(u_t = 0|\widetilde{\boldsymbol{v}} \text{ et } \widetilde{\boldsymbol{w}}) + \boldsymbol{P}(u_t = 1|\widetilde{\boldsymbol{v}} \text{ et } \widetilde{\boldsymbol{w}}) = 1$ et $(a, b) : a \xrightarrow{i} b$ représente tous les changements d'état interne du codeur effectué en entrant un i dans celui-ci.

2.4 Entrelacement

De part leur construction, les codes convolutifs sont très sensibles aux erreurs arrivant par rafales, c'est-à-dire concernant un ensemble de bit proches les uns des autres. Dans ce cas, même si le niveau de bruit sur l'ensemble du mot reçu est relativement faible les algorithmes de décodage ne retrouvent pas nécessairement l'information transmise. Afin de contrer ce phénomène, les mots de codes sont entrelacés. Autrement dit, les bits d'un même mot de code sont permutés entre eux avant la transmission sur le canal. La situation est représentée sur la figure 2.9. Si le mot observé en sortie de canal possède des zones bruitées par des rafales d'erreurs, en appliquant la permutation inverse ces erreurs seront réparties sur l'ensemble du mot. Les algorithmes de décodage des codes convolutifs peuvent alors être appliqués afin de retrouver l'information. L'entrelacement permet donc d'augmenter très simplement la capacité de correction des codes convolutifs contre les erreurs par rafales. Ce principe d'entrelacement est utilisé dans la construction des turbo-codes.

2.5 Turbo-codes

Les turbo-codes sont une famille de codes correcteurs d'erreurs inventés par Claude Berrou et Alain Glavieux en 1993 [10]. Grâce à leurs performances approchant la limite de Shannon, à la facilité de leur implémentation ainsi qu'à la rapidité de leurs codage et décodage, ils se sont imposés, dès les années 2000, dans de nombreux standards de télécommunication. Aujourd'hui, ils sont très largement utilisés. Par exemple, les normes IEEE 802.16 [67], 802.20 [69] et

Algorithme 1 : Algorithme BCJR (décodage des codes convolutifs)

Entrée :
$\widetilde{\boldsymbol{v}}$ et $\widetilde{\boldsymbol{w}}$ les données observées en sortie de canal
s l'état initial du codeur
Sortie :
$\widetilde{\boldsymbol{u}}$ l'information la plus vraisemblablement émise

---

Étape 1 :
Pour t de 1 à N
Pour (a, b) parcourant les 2×2^m changements d'état interne du codeur
$i, j \leftarrow$ sorties émises par le codeur en passant de l'état a à l'état b
$\gamma_{a,b}^t \leftarrow \boldsymbol{P}(v_t = i|\widetilde{v}_t)\boldsymbol{P}(w_t = j|\widetilde{w}_t)$
Étape 2 :
$\alpha_s^0 \leftarrow 1$
Pour tout état $a \neq s$: $\alpha_a^0 \leftarrow 0$
Pour t de 1 à N
$somme \leftarrow 0$
Pour a parcourant les 2^m états internes du codeur
$\alpha_a^t \leftarrow \sum_{b=0}^{2^m-1} \alpha_b^{t-1}\gamma_{b,a}^t$
$somme \leftarrow somme + \alpha_a^t$
Pour a parcourant les 2^m états internes du codeur
$\alpha_a^t \leftarrow \alpha_a^t / somme$ $\quad //\alpha_a^t = \boldsymbol{P}(e_t = a|\widetilde{\boldsymbol{v}}_{1..t}, \widetilde{\boldsymbol{w}}_{1..t})$
Étape 3 :
Pour tout état a : $\beta_a^N \leftarrow 1/(2^m - 1)$
Pour t de $N-1$ à 0
$somme \leftarrow 0$
Pour a parcourant les 2^m états internes du codeur
$\beta_a^t \leftarrow \sum_{b=0}^{2^m-1} \beta_b^{t+1}\gamma_{a,b}^{t+1}$
$somme \leftarrow somme + \beta_a^t$
Pour a parcourant les 2^m états internes du codeur
$\beta_a^t \leftarrow \beta_a^t / somme$ $\quad //\beta_a^t = \boldsymbol{P}(e_t = a|\widetilde{\boldsymbol{v}}_{t+1..N}, \widetilde{\boldsymbol{w}}_{t+1..N})$
Étape 4 :
Pour t de $N-1$ à 0
$\delta_0^t \leftarrow \sum_{(a,b):a \overset{0}{\to} b} \alpha_a^{t-1}\gamma_{a,b}^t\beta_b^t$
$\delta_1^t \leftarrow \sum_{(a,b):a \overset{1}{\to} b} \alpha_a^{t-1}\gamma_{a,b}^t\beta_b^t$
Si $\delta_0^t/(\delta_0^t + \delta_1^t) > 0.5$ $\quad //\delta_0^t/(\delta_0^t + \delta_1^t) = \boldsymbol{P}(u_t = 0|\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{w}})$
$\widetilde{u_t} \leftarrow 0$
Sinon
$\widetilde{u_t} \leftarrow 1$
Retourner $\widetilde{\boldsymbol{u}}$
En fonction de l'utilisation de cet algorithme il peut être plus intéressant de retourner les probabilités $\boldsymbol{P}(u_t = 0|\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{w}})$ données par $\delta_0^t/(\delta_0^t + \delta_1^t)$.

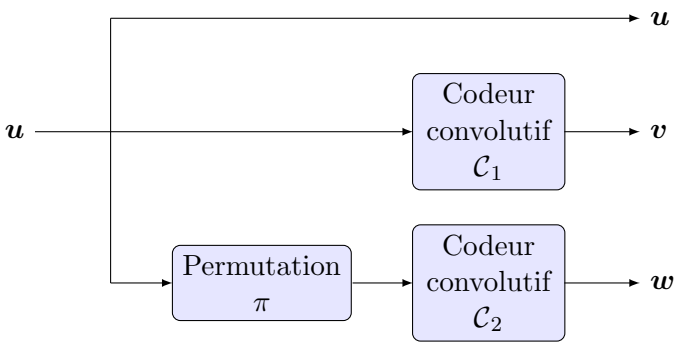


FIGURE 2.10 – Schéma de codage d'un turbo-code parallèle

802.22 [70] font appel à des turbo-codes, c'est également le cas des normes ETSI GMR-1 [47] et DVB-RCS2 [40] ou encore de la norme CCSDS 131 [15].

Les turbo-codeurs sont des codes en bloc composés de deux codeurs convolutifs et d'un entrelaceur (c'est-à-dire une permutation). Il existe deux types de turbo-codes, les turbo-codes parallèles et les turbo-codes série. Ces codes diffèrent par leur schéma de codage et doivent leurs noms à la façon dont sont connectés les codeurs convolutifs dans ces schémas.

2.5.1 Turbo-codes parallèles

Les turbo-codes parallèles sont très largement utilisés en pratique. Le codage s'effectue en suivant le schéma représenté sur la figure 2.10. Les deux codeurs convolutifs sont supposés de longueur 2 et de dimension 1. Seule la sortie de redondance de ces codeurs est conservée pour former le mot de code, ces codeurs étant systématiques et récursifs. Les vecteurs $\boldsymbol{u}$, $\boldsymbol{v}$ et $\boldsymbol{w}$ sont tous trois de même taille, notée N. Le code ainsi construit est donc de longueur $3N$, de dimension N et de rendement $1/3$.

En sortie de canal, trois vecteurs bruités $\widetilde{\boldsymbol{u}}$, $\widetilde{\boldsymbol{v}}$ et $\widetilde{\boldsymbol{w}}$ sont observés. Afin de retrouver à partir de ces vecteurs l'information $\boldsymbol{u}$, le décodage suit le schéma représenté sur la figure 2.11. Deux décodeurs convolutifs sont placés en série afin d'utiliser simultanément les informations données par chacun d'eux. Il s'agit donc d'un décodage itératif au sens où la boucle enchaînant décodage de $\mathcal{C}_1$, application de π, décodage de $\mathcal{C}_2$ et désentrelacement est répétée plusieurs fois. Les décodeurs convolutifs correspondent, généralement, à l'application de l'algorithme BCJR présenté précédemment. La rapidité de l'algorithme BCJR, comme celle des autres algorithmes de décodage de codes convolutifs, dépend du nombre de mémoires du codeur. Pour des raisons de complexité mais aussi de performances, les codes convolutifs utilisés en pratique possèdent au maximum 4 mémoires. En effet, ce doit être des codes convolutifs réducteurs d'erreurs et non correcteurs d'erreurs.

Pour avoir plus de liberté sur le rendement du code, tout en conservant le même codeur, les turbo-codes peuvent être poinçonnés. Le poinçonnage consiste à ne pas transmettre certains bits appartenant aux vecteurs des sorties non systématiques. Par exemple, en transmettant un bit sur deux des vecteurs $\boldsymbol{v}$ et $\boldsymbol{w}$, le turbo-code obtenu est de rendement $1/2$. En utilisant différents motifs de poinçonnage, des turbo-codes de rendements compris entre $1/2$ et $5/6$ sont utilisés en pratique. De plus, dans les normes de télécommunication, les codeurs convolutifs

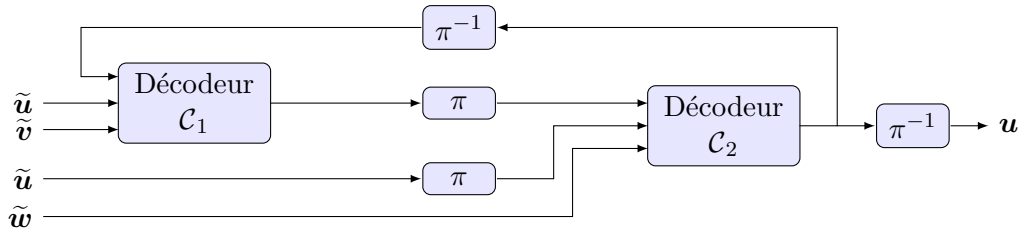


FIGURE 2.11 – Schéma de décodage d'un turbo-code parallèle

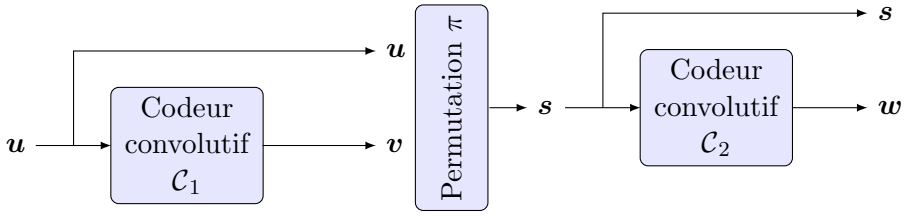


FIGURE 2.12 – Schéma de codage d'un turbo-code série

ne sont pas nécessairement des codes $(2, 1)$, la norme IEEE 802.20 [69] utilise par exemple des codeurs convolutifs systématiques et récursifs de dimension 1 et de longueur 3.

2.5.2 Turbo-codes série

Un autre type de turbo-code existe, il s'agit des turbo-codes série, comme leur nom l'indique, les deux codeurs convolutifs sont placés en série, c'est-à-dire appliqués l'un après l'autre. Le schéma de codage des turbo-code série est représenté sur la figure 2.12.
Comme pour les turbo-codes parallèles, les deux codeurs convolutifs sont supposés systématiques et récursifs, seules les sorties non systématiques sont conservées. Il s'agit généralement de codeurs de type $(2, 1)$, le turbo-code ainsi construit est alors de rendement $1/4$. Ce rendement peut être modulé grâce à l'application d'un poinçonnage sur les sorties des codeurs convolutifs.

Chapitre 3

Les codes LDPC

3.1 Introduction

Les codes LDPC (Low-Density Parity-Check) sont des codes correcteurs très utilisés aujourd'hui. Ils ont été inventés en 1962 par Gallager [53], mais n'ont pas été utilisés à l'époque. Ils ont alors été oubliés pendant une trentaine d'années, avant d'être redécouverts en 1996 par MacKay [98] puis réellement utilisés. Les codes LDPC sont des codes en bloc linéaires, à l'origine binaires, ils ont depuis été généralisés au cas non binaire [30]. Nous nous intéresserons uniquement au cas binaire, cependant il peut être noté que les codes LDPC non binaires sont de plus en plus étudiés, ils commencent à être utilisés dans certaines normes de télécommunication telles que la norme CCSDS 231.1-O-1 d'avril 2015 [17].

Comme leur nom l'indique, ils sont caractérisés par une matrice de parité $\boldsymbol{\mathcal{H}}$ très creuse. Autrement dit, un code est dit LDPC s'il admet une matrice de parité $\boldsymbol{\mathcal{H}}$ qui possède très peu de coefficients non nuls.

Les codes LDPC ont de très bonnes performances, il est même conjecturé que ces codes peuvent atteindre la capacité de n'importe quel canal binaire symétrique sans mémoire lorsque la distribution de degrés du code LDPC est bien choisie (c'est-à-dire le nombre de coefficients non-nuls par ligne et par colonne dans la matrice de parité). En 2013, cette conjecture a été démontrée par Kudekar, Richardson et Urbanke [79] pour une sous-famille de codes LDPC, les codes LDPC spatialement couplés.
De plus, ils atteignent ces performances avec un algorithme de décodage itératif peu complexe. Les codes linéaires aléatoires ont eux aussi de très bonnes performances, mais celles-ci sont atteintes au maximum de vraisemblance, or cet algorithme n'est pas réalisable en pratique. Ceci explique l'intérêt des codes LDPC en pratique.

Dans ce chapitre, nous rappelons quelques notions concernant les codes LDPC telles que leur codage et leur décodage.

Notations. Nous notons $\boldsymbol{\mathcal{H}}$ une matrice de parité (binaire) d'un code LDPC de longueur N, de dimension K et admettant la matrice $\boldsymbol{\mathcal{G}}$ pour matrice génératrice. Le rendement de ce code est noté $R = \frac{K}{N}$.

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

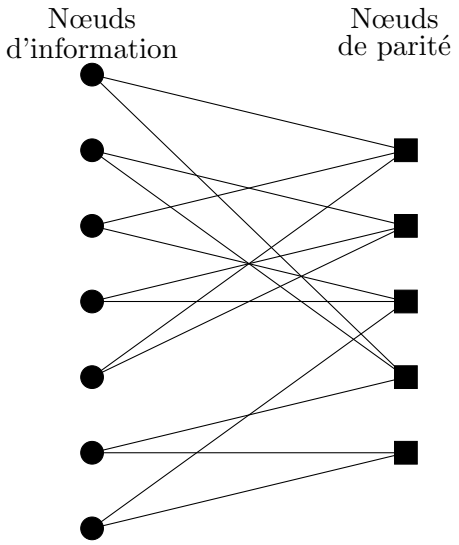


FIGURE 3.1 – Une matrice de parité $\boldsymbol{\mathcal{H}}$ et son graphe de Tanner

3.2 Graphes de Tanner

A toute matrice de parité $\boldsymbol{\mathcal{H}}$, un graphe biparti, dit graphe de Tanner, peut lui être associé. Ce graphe est utilisé pour le décodage des codes LDPC.

Le graphe de Tanner associé à la matrice de parité $\boldsymbol{\mathcal{H}}$ possède N nœuds (ou sommets), dit nœuds d'information, représentant les bits du mot de code et $N-K$ nœuds de parité qui représentent les équations de parité. Le nœud d'information représentant le i-ème bit est relié au nœud de parité représentant la j-ème équation de parité si et seulement si le coefficient à la i-ème colonne et j-ème ligne de $\boldsymbol{\mathcal{H}}$ est non nul.

Exemple. *Une matrice de parité $\boldsymbol{\mathcal{H}}$ et le graphe de Tanner qui lui est associé sont représentés sur la figure 3.1. Sur cette figure, et sur les suivantes, les nœuds d'information sont notés par des ronds et les nœuds de parité par des carrés.*

Afin que le code LDPC considéré ait de bonnes performances de correction, le graphe de Tanner associé à sa matrice de parité ne doit pas contenir de petits cycles (et en particulier de longueur 4). En effet, la présence de petits cycles détériore de manière significative les performances du décodage itératif.

Graphe de Tanner et mots de code. Par définition d'une matrice de parité, un mot $\boldsymbol{c}$ est un mot de code si et seulement si $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \boldsymbol{0}$. Graphiquement, cette propriété devient : un mot $\boldsymbol{c} = (c_1, \ldots, c_N)$ est un mot de code si et seulement si en associant à chacun des nœuds d'information i la valeur de c_i alors, pour tous les nœuds de parité, la somme modulo 2 des valeurs associées à ses voisins est nulle. (Pour rappel, deux sommets sont voisins s'il existe une arête reliant ces deux sommets.)

3.3 Codage et décodage des codes LDPC

Dans le cas général, le codage d'un code LDPC s'effectue par multiplication matricielle, grâce à la matrice génératrice $\boldsymbol{\mathcal{G}}$. Au mot d'information $\boldsymbol{u}$ est alors associé le mot de code $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{\mathcal{G}}$.

Par définition, la matrice de parité d'un code LDPC est creuse, cependant ce n'est pas nécessairement le cas de la matrice génératrice. Le coût du codage est alors quadratique, or, en pratique, il est généralement souhaitable d'avoir un algorithme de codage de complexité linéaire. Si le code LDPC considéré n'est pas choisi aléatoirement, par exemple s'il possède une matrice de parité structurée, l'algorithme de codage est adapté et peut devenir de complexité linéaire. En pratique, les codes LDPC sont toujours structurés, comme nous le verrons par la suite. Nous détaillerons alors les méthodes de codage adaptées à la structure de la matrice de parité.

Concernant le décodage, plusieurs algorithmes existent tels que "Bit Flipping", "Sum-Product" et "Min-Sum". Ces algorithmes sont utilisables quel que soit le code LDPC, qu'il soit structuré ou non. Comme pour le codage, des méthodes adaptées à la structure de la matrice de parité ont été développées afin d'optimiser les performances des codes utilisés.

Nous présentons dans ce chapitre l'algorithme de décodage "Sum-Product". Cet algorithme est utilisé lorsque les données reçues sont souples, c'est-à-dire lorsque nous disposons de la probabilité de valoir 0 ou 1 pour chacun des bits. C'est le cas lorsqu'une démodulation souple est effectuée, cela permet de conserver le plus d'information possible. En pratique, une telle démodulation est généralement effectuée. Par un processus itératif ces probabilités vont évoluer jusqu'à converger. Le mot de code en sera alors déduit.

Pour cet algorithme, deux calculs de probabilité sont utilisés :

Proposition 3.1. *Soient $v_1, \ldots, v_n$ des valeurs binaires indépendantes, et soit $p_i = \boldsymbol{P}(v_i = 0)$. La probabilité que la somme modulo 2 des v_i soit nulle est :*

$$\boldsymbol{P}\left(\sum_{i=1}^{n} v_i = 0 | p_1, \ldots, p_n\right) = \frac{1 + \prod_{i=1}^{n}(2p_i - 1)}{2}$$

Proposition 3.2. *Soient v une valeur binaire telle que $\boldsymbol{P}(v = 0) = 1/2$. Soient $E_1, \ldots, E_n$ un ensemble de n événements conditionnellement indépendants par rapport à v, autrement-dit : $\boldsymbol{P}(E_1, \ldots, E_n | v = b) = \boldsymbol{P}(E_1|v = b) \ldots \boldsymbol{P}(E_n|v = b)$ pour $b = 0$ et $b = 1$. Et soient les probabilités $p_1 = \boldsymbol{P}(v = 0|E_1), p_2 = \boldsymbol{P}(v = 0|E_2), \ldots, p_n = \boldsymbol{P}(v = 0|E_n)$ alors*

$$\boldsymbol{P}(v = 0 | E_1, \ldots, E_n) = \frac{\prod_{k=1}^{n} p_k}{\prod_{k=1}^{n} p_k + \prod_{k=1}^{n}(1 - p_k)}$$

Les démonstrations de ces deux propositions peuvent, par exemple, être trouvées dans la thèse de Mathieu Cluzeau [22], pages 65 et 66.

Notations.
— Soit le message reçu $\boldsymbol{r} = (r_1, \ldots, r_N)$ et $\boldsymbol{c} = (c_1, \ldots, c_N)$ le mot de code émis.

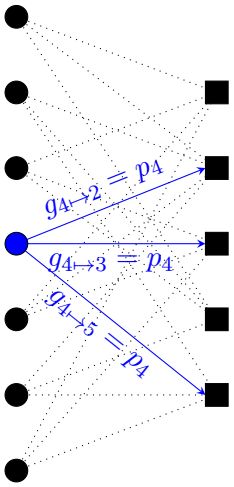


FIGURE 3.2 – Initialisation

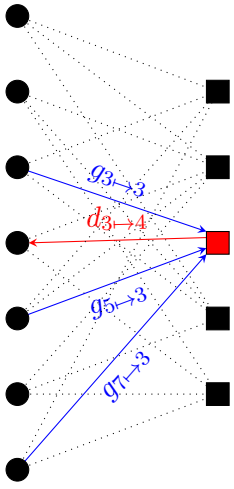


FIGURE 3.3 – Étape 1

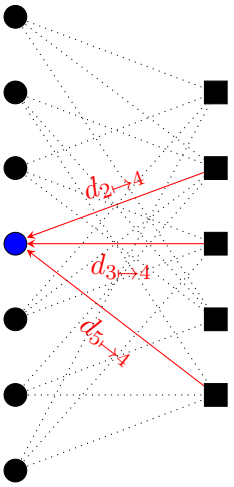


FIGURE 3.4 – Décision

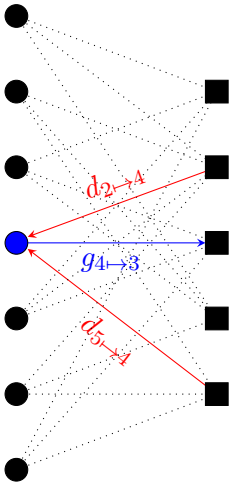


FIGURE 3.5 – Étape 2

— Pour tout $i \leq N$, la probabilité $p_i = \boldsymbol{P}(c_i = 0|r_i)$ est supposée connue (son calcul dépend du canal de communication utilisé comme expliqué dans le chapitre 1).
— $\mathcal{E}_j$ représente l'ensemble des indices intervenant dans la j-ème équation de parité : $\mathcal{E}_j = \{i$ tels que le i-ème nœud d'information est voisin du j-ème nœud de parité$\}$.
— $\mathcal{F}_i$ représente l'ensemble des équations de parité faisant intervenir le i-ème bit : $\mathcal{F}_i = \{j$ tels que le i-ème nœud d'information est voisin du j-ème nœud de parité$\}$
— Lors du décodage, les nœuds du graphe de Tanner vont s'échanger des probabilités. Notons $g_{i\mapsto j}$ la probabilité transmise par le i-ème nœud d'information au j-ème nœud de parité (ce transfert se fait donc de gauche à droite sur le graphe de Tanner). $d_{j\mapsto i}$ représente la probabilité transmise par le j-ème nœud de parité au i-ème nœud d'information (sur le graphe de Tanner, cet échange s'effectue de droite à gauche).

L'algorithme de décodage "Sum-Product" est itératif. Avant de débuter les itérations, une phase d'initialisation est nécessaire, puis à chaque tour de boucle deux étapes sont effectuées. Une étape intermédiaire de décision permet d'arrêter ou non le décodage.

Initialisation. Au début du décodage, à chaque nœud d'information i est associée la probabilité $p_i = \boldsymbol{P}(c_i = 0|r_i)$. Ces nœuds doivent transmettre cette information aux nœuds de parité auxquels ils sont reliés, afin que ceux-ci déterminent quelles sont les équations de parité qui sont vérifiées. On a donc lors de la phase d'initialisation : $g_{i\mapsto j} = p_i$ pour tout nœud d'information i et tout nœud de parité j adjacent à i (la figure 3.2 illustre cette phase d'initialisation).

Étape 1 : des nœuds de parité aux nœuds d'information. Le nœud de parité j transmet au nœud d'information i la probabilité $d_{j\mapsto i}$ que c_i soit nul connaissant toutes les informations reçues par j de la part de ses autres voisins. Autrement dit, j transmet à i la probabilité que c_i soit nul, sachant que la j-ème équation de parité est satisfaite et connaissant la probabilité d'être nul de chacun des autres bits intervenant dans cette équation de parité. La situation est représentée par la figure 3.3. Afin de déterminer $d_{j\mapsto i}$ la proposition 3.1 est utilisée :

$$\begin{aligned} d_{j\mapsto i} &= \boldsymbol{P}(c_i = 0 | g_{k\mapsto j}, \forall k \in \mathcal{E}_j, k \neq i) \\ &= \boldsymbol{P}\left(\sum_{k\in\mathcal{E}_j, k\neq i} c_k = 0 | g_{k\mapsto j}, \forall k \in \mathcal{E}_j, k \neq i \right) \\ &= \frac{1 + \prod_{k\in\mathcal{E}_j, k\neq i}(2g_{k\mapsto j} - 1)}{2} \end{aligned}$$

Décision. À cette étape l'algorithme détermine le mot le plus probablement émis en fonction des probabilités calculées jusque là. Pour cela, la probabilité d'être nul de chacun des bits du message est définie en fonction de sa probabilité initiale et de toutes les probabilités reçues par le nœud d'information le représentant. L'illustration de cette étape est faite par la figure 3.4. Un ensemble de probabilités portant sur la même information est donc connue, la proposition 3.2 permet alors de déterminer la probabilité recherchée :

$$\boldsymbol{P}(c_i = 0 | p_i, d_{k\mapsto i}, \forall k \in \mathcal{F}_i) = \frac{p_i \prod_{k\in\mathcal{F}_i} d_{k\mapsto i}}{p_i \prod_{k\in\mathcal{F}_i} d_{k\mapsto i} + (1 - p_i) \prod_{k\in\mathcal{F}_i}(1 - d_{k\mapsto i})}$$

Grâce à ces probabilités, le mot (binaire) $\tilde{\boldsymbol{c}} = (\tilde{c}_1, \ldots, \tilde{c}_N)$ le plus probablement émis en est déduit : $\forall i \in \{1, \ldots, N\}$,

$$\begin{cases} \tilde{c}_i = 0 & \text{si } \boldsymbol{P}(c_i = 0 | p_i, d_{k\mapsto i}, \forall k \in \mathcal{F}_i) > 0.5 \\ \tilde{c}_i = 1 & \text{sinon} \end{cases}$$

Il faut ensuite vérifier que le mot $\tilde{\boldsymbol{c}}$ est bien un mot du code LDPC. Cette vérification se fait à l'aide du produit $\boldsymbol{\mathcal{H}}\tilde{\boldsymbol{c}}^T$, si celui-ci est nul alors $\tilde{\boldsymbol{c}}$ est un mot de code, le décodage est donc terminé, dans le cas contraire, il faut poursuivre le décodage.

Étape 2 : des nœuds d'information aux nœuds de parité. Si une nouvelle itération est nécessaire afin d'obtenir un mot de code, les nœuds d'information font parvenir aux nœuds de parité de nouvelles probabilités. Chaque nœud d'information i transmet à chacun de ses voisins j la probabilité $g_{i\mapsto j}$ que le bit qu'il représente soit nul, connaissant sa probabilité initiale p_i ainsi que les probabilités $d_{k\mapsto i}$ qu'il a reçu de la part de ses autres voisins ($k \in \mathcal{F}_i$, $k \neq j$). Comme pour l'étape de décision, la proposition 3.2 est utilisée :

$$\begin{aligned} g_{i\mapsto j} &= \boldsymbol{P}(c_i = 0 | p_i, d_{k\mapsto i}, k \in \mathcal{F}_i, k \neq j) \\ &= \frac{p_i \prod_{k\in\mathcal{F}_i, k\neq j} d_{k\mapsto i}}{p_i \prod_{k\in\mathcal{F}_i, k\neq j} d_{k\mapsto i} + (1 - p_i) \prod_{k\in\mathcal{F}_i, k\neq j}(1 - d_{k\mapsto i})} \end{aligned}$$

Algorithme de décodage. L'algorithme de décodage "Sum-Product" est finalement donné par l'algorithme 2. Afin qu'il se termine, un nombre maximal d'itérations à effectuer est donné en entrée.

Algorithme 2 : DÉCODAGE D'UN CODE LDPC

Entrée :

$\boldsymbol{\mathcal{H}}$ la matrice de parité du code LDPC utilisé

$p_i = \boldsymbol{P}(c_i = 0|r_i)$, $\forall i \in \{1, \ldots, N\}$

$nbIterMax$ le nombre maximal d'itérations à effectuer

Sortie :

$$\begin{cases} \tilde{\boldsymbol{c}} = (\tilde{c}_1, \ldots, \tilde{c}_N) & \text{si le décodage a réussi :} \\ & \text{le mot de code le plus probablement émis} \\ \text{“}échec\text{”} & \text{sinon} \end{cases}$$

$nbIter \leftarrow 0$

Initialisation :

$g_{i \mapsto j} \leftarrow p_i$, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \mathcal{F}_i$

Tant que nbIter $<$ nbIterMax **faire**

Étape 1 :

$$d_{j \mapsto i} \leftarrow \frac{1+\prod_{k \in \mathcal{E}_j, k \neq i}(2 g_{k \mapsto j} - 1)}{2}, \forall j \in \{1, \ldots, N-K\}, \forall i \in \mathcal{E}_j$$

Décision : $\forall i \in \{1, \ldots, N\}$

$$\begin{cases} \tilde{c}_i \leftarrow 0 & \text{si } \frac{p_i \prod_{k \in \mathcal{F}_i} d_{k \mapsto i}}{p_i \prod_{k \in \mathcal{F}_i} d_{k \mapsto i} + (1-p_i) \prod_{k \in \mathcal{F}_i} (1 - d_{k \mapsto i})} > 0.5 \\ \tilde{c}_i \leftarrow 1 & \text{sinon} \end{cases}$$

Si $\boldsymbol{\mathcal{H}} \tilde{\boldsymbol{c}}^T = 0$

Retourner $\tilde{\boldsymbol{c}}$

Étape 2 :

$$g_{i \mapsto j} \leftarrow \frac{p_i \prod_{k \in \mathcal{F}_i k \neq j} d_{k \mapsto i}}{p_i \prod_{k \in \mathcal{F}_i k \neq j} d_{k \mapsto i} + (1-p_i) \prod_{k \in \mathcal{F}_i, k \neq j} (1 - d_{k \mapsto i})}, \forall i \in \{1, \ldots, N\}, \forall j \in \mathcal{F}_i$$

$nbIter \leftarrow nbIter + 1$

Retourner “*échec*”

Chapitre 4

Les codes LDPC binaires normés

De nombreuses normes de télécommunication utilisent des codes LDPC et elles sont en constante augmentation. Afin de développer une méthode de reconnaissance des codes LDPC adaptée aux codes utilisés en pratique, une étude de toutes (ou presque) les normes utilisant des codes LDPC a été effectuée. La définition des codes LDPC est très large, connaître les codes utilisés permet d'avoir une vue sur l'ensemble des paramètres tels que les longueurs mais également les poids des équations de parité. Ce sont ces deux paramètres qui influeront sur la méthode de reconnaissance présentée dans le chapitre 6.

L'étude des normes de télécommunication faisant appel à des codes LDPC a permis de mettre en évidence que tous les codes utilisés dans la pratique (de façon normalisée) sont systématiques mais surtout qu'ils possèdent une matrice de parité extrêmement structurée. Ce chapitre est consacré à ces codes LDPC binaires normés, dans chacun des cas, la forme de la matrice de parité est détaillée ainsi qu'une méthode de codage adaptée à cette structure. Les longueurs et dimensions de ces codes se trouvent en annexe A.

Les matrices de parité normées peuvent être classées en deux groupes en fonction de si elles ont, ou non, une structure convolutive. Elles peuvent également être regroupées en fonction de leur quasi-cyclicité. Ces deux caractéristiques pouvant être satisfaites simultanément, nous avons décidé de présenter, dans un premier temps, les matrices avec une structure convolutive puis de les classer en sous-groupes selon qu'elles soient ou non quasi-cycliques. Le même partitionnement en sous-groupes est effectué, dans un second temps, sur l'ensemble des matrices qui ne possèdent pas de structure convolutive.

Dans les normes, les mots de codes sont parfois poinçonnés après l'encodage, c'est-à-dire que certains bits, à des positions fixées, ne sont pas transmis. Les performances des codes LDPC permettent de retrouver, lors du décodage, la valeur de ces bits. Les matrices de parité présentées dans les sections suivantes correspondent aux matrices de parité des codes non poinçonnés.

4.1 Représentations et définitions

Afin de simplifier les représentations, les matrices de parité sont illustrées de façon binaire. Sur ces schémas, un petit carré noir indique un bit non nul et un petit carré blanc un bit nul.

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

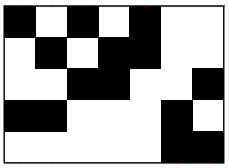
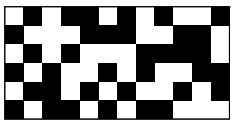


FIGURE 4.1 – Représentations binaire et graphique d'une même matrice de parité



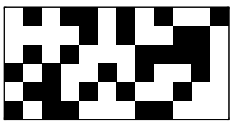


FIGURE 4.2 – Une matrice régulière (à gauche) et une matrice irrégulière (à droite)

Exemple. *La matrice représentée à gauche sur la figure 4.1 a pour illustration la matrice de droite.*

Dans la suite de ce chapitre les définitions suivantes sont utilisées :

Définition 4.1 (Matrice régulière)**.** *Une matrice est dite régulière si elle possède un nombre constant d'éléments non nul sur chacune de ses lignes et chacune de ses colonnes. Une matrice qui n'est pas régulière est dite irrégulière.*

Exemple. *Sur la figure 4.2 sont représentées une matrice régulière (à gauche) et une matrice irrégulière (à droite).*

Définition 4.2 (Code LDPC régulier)**.** *Un code LDPC est dit régulier s'il possède une matrice de parité régulière. Dans le cas contraire, le code LDPC est dit irrégulier.*

Définition 4.3 (Matrice avec une structure convolutive)**.** *Une matrice est dite avec une structure convolutive si les coefficients non-nuls de ses dernières colonnes forment une double diagonale. (La diagonale supérieure commençant sur la première ligne de la matrice.)*

Le terme de "structure convolutive" est lié au codage de ces codes LDPC. En effet, comme expliqué dans la section suivante, ils peuvent être codés en utilisant un codeur convolutif.

Exemple. *Sur la figure 4.3 sont représentées trois matrices, deux avec une structure convolutive et une matrice sans structure convolutive.*

Définition 4.4 (Code LDPC avec une structure convolutive)**.** *Un code LDPC est dit avec une structure convolutive s'il possède une matrice de parité avec une structure convolutive.*

Définition 4.5 (Matrice cyclique)**.** *Une matrice carrée est dite cyclique si ses lignes sont invariantes à permutation circulaire près.*

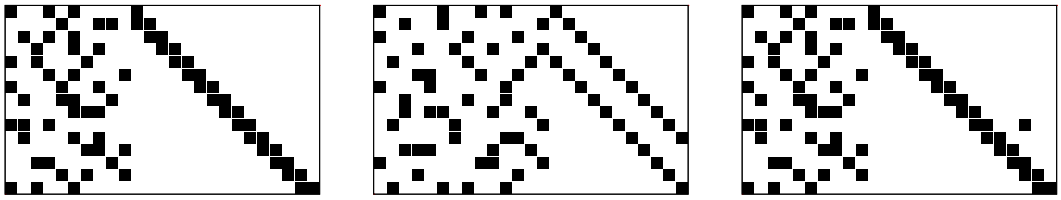


FIGURE 4.3 – Deux matrices avec une structure convolutive (à gauche et au centre) et une matrice sans structure convolutive (à droite)

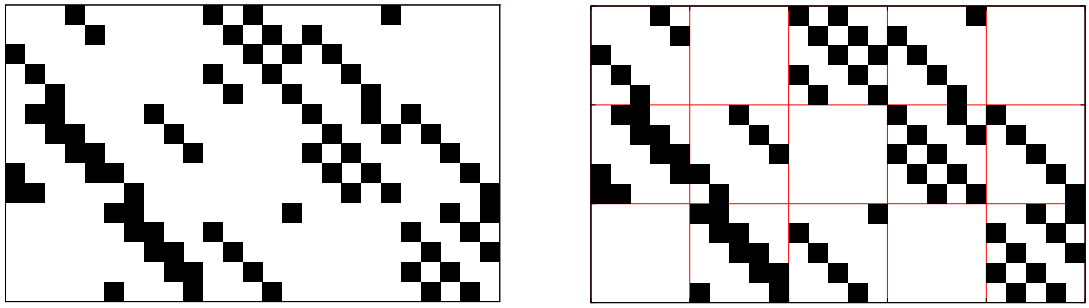


FIGURE 4.4 – Une matrice quasi-cyclique et la même matrice avec son partitionnement en sous-matrices cycliques

Définition 4.6 (Matrice quasi-cyclique)**.** *Une matrice est dite quasi-cyclique d'ordre Z si elle peut être partitionnée en sous-matrices cycliques de taille $Z \times Z$.*

Exemple. *Sur la figure 4.4 est représentée une matrice quasi-cyclique.*

Définition 4.7 (Code LDPC quasi-cyclique)**.** *Un code LDPC est dit quasi-cyclique s'il possède une matrice de parité quasi-cyclique.*

4.2 Codage des codes LDPC avec une structure convolutive

Pour les codes LDPC avec une structure convolutive, comme pour tout ceux utilisés en pratique, le codage est systématique. Le mot de code $\boldsymbol{c}$ associé au mot d'information $\boldsymbol{u} = (u_1, \ldots, u_K)$ est donc de la forme $\boldsymbol{c} = (u_1, \ldots, u_K, c_{K+1}, \ldots, c_N)$.

Si la structure de double diagonale de la matrice de parité est présente sur les $N - K$ dernières colonnes de celle-ci, le codage peut être schématisé par la figure 4.5. Il s'explique grâce au graphe de Tanner que l'on peut déplier afin de placer à droite tous les nœuds d'information représentant des bits de parité. La structure convolutive est alors mise en évidence par la régularité des arêtes reliant les nœuds de parité aux nœuds d'information qui ont été déplacés (voir figure 4.6). Afin que les équations de parité soient satisfaites, le bit de parité c_i dépend du bit de parité c_{i-1}, d'où l'utilisation du terme de structure convolutive.

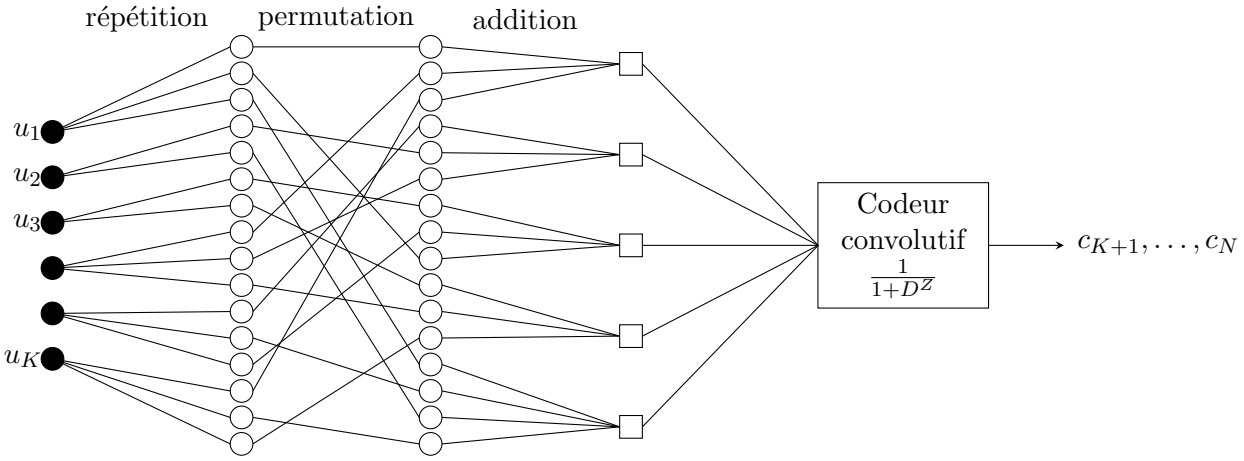


FIGURE 4.5 – Schéma de codage des codes LDPC avec une structure convolutive

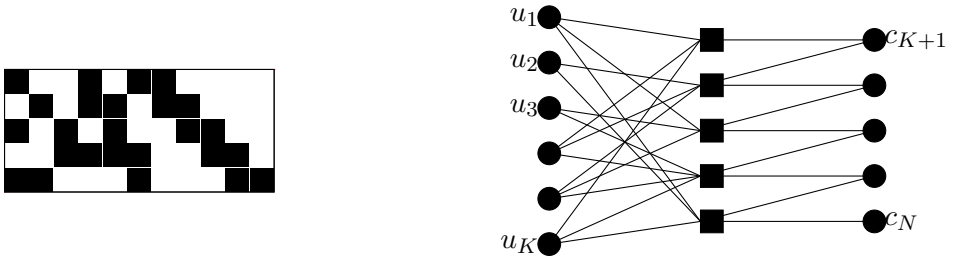


FIGURE 4.6 – Matrice de parité avec une structure convolutive et son graphe de Tanner déplié

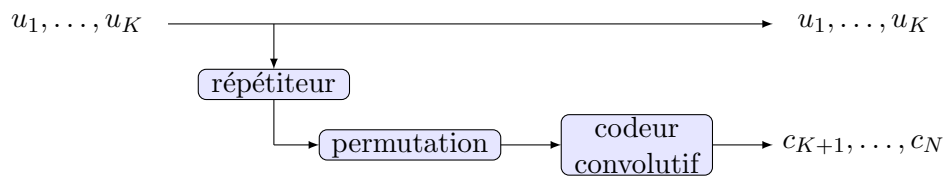


FIGURE 4.7 – Étapes de codage des codes LDPC avec structure convolutive

Afin de déterminer la valeur des $N - K$ bits de parité $(c_{K+1}, \ldots, c_N)$, différentes étapes sont effectuées. La première correspond à un répétiteur, chaque bit d'information est dupliqué un certain nombre de fois, ils sont ensuite permutés afin de calculer des valeurs intermédiaires. Ces valeurs sont ensuite entrées dans un codeur convolutif défini par $(1, \frac{1}{1+D^Z})$ où Z est l'écart entre les deux diagonales de la structure convolutive de $\boldsymbol{\mathcal{H}}$. Les bits de redondance émis par ce codeur convolutif correspondent aux bits de redondance recherchés.

Remarque. *Pour des matrices quasi-cycliques avec une structure convolutive, Z est également l'ordre de quasi-cyclicité.*

Il peut être remarqué que le codeur convolutif défini par $(1, \frac{1}{1+D^Z})$ est un accumulateur. D'où les noms donnés à ces codes dans la littérature :

— RA (repeat accumulate) lorsque la sous matrice $\boldsymbol{\mathcal{A}}$, composée des K premières colonnes de la matrice de parité $\boldsymbol{\mathcal{H}}$, est régulière.
— IRA (irregular repeat accumulate) si la sous matrice $\boldsymbol{\mathcal{A}}$, composée des K premières colonnes de $\boldsymbol{\mathcal{H}}$, est irrégulière.
— S-IRA (structured-irregular repeat accumulate) ou QC-IRA (quasi-cyclique irregular repeat accumulate) si la matrice de parité $\boldsymbol{\mathcal{H}}$ est également quasi-cyclique.

De plus, ces trois étapes de codage sont similaires au codage des turbo-codes, comme représenté sur la figure 4.7. Seul le répétiteur est ajouté par rapport au schéma classique du codage des turbo-codes (en considérant uniquement la première et la dernière sortie du turbo-codeur).

Plus précisément, chaque bit d'information est dupliqué autant de fois qu'il intervient dans une équation de parité (le bit u_i est répété x fois, où x est le poids de la i-ème colonne de $\boldsymbol{\mathcal{H}}$). Ces bits sont ensuite permutés afin de les ordonner de sorte que des bits intervenant dans une même équation de parité soient consécutifs (en premier, sont donc placés une copie de chacun des bits impliqués dans la première équation de parité, puis une copie de ceux intervenant dans la seconde équation de parité et ainsi de suite). Les sommes, modulo 2, des bits d'information intervenant dans chacune des équations de parité sont alors déterminées. Ce sont ces valeurs qui sont codées grâce au codeur convolutif.

Sur l'exemple de la figure 4.6, les deux diagonales sont collées, $Z = 1$, le code convolutif utilisé est donc défini par $(1, \frac{1}{1+D})$. Lorsque Z est supérieur à 1, le codage fait toujours appel à un codeur convolutif, mais celui-ci peut également être parallélisé, ce qui le rend plus rapide. Le facteur de parallélisation est égal à Z, la figure 4.8 illustre cette situation pour $Z = 3$ à l'aide d'une matrice de parité et de son graphe de Tanner déplié (les nœuds représentant des bits de parité sont placés à droite). Les différentes couleurs indiquent les bits de parité qui sont liés entre eux. Le schéma de codage correspondant est celui de la figure 4.9.

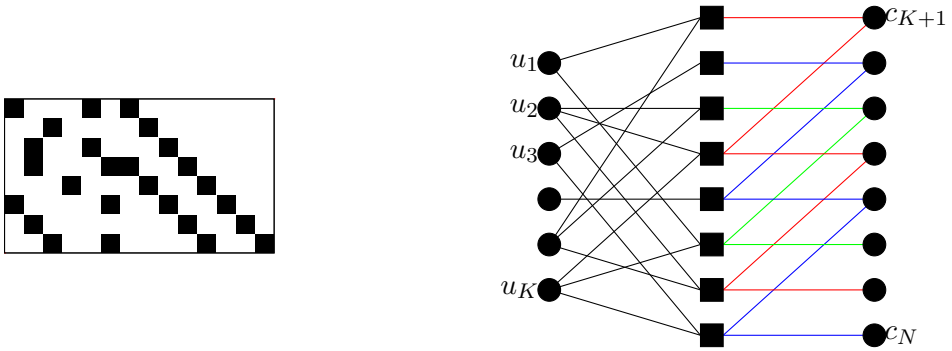


FIGURE 4.8 – Matrice de parité avec une structure convolutive et son graphe de Tanner déplié

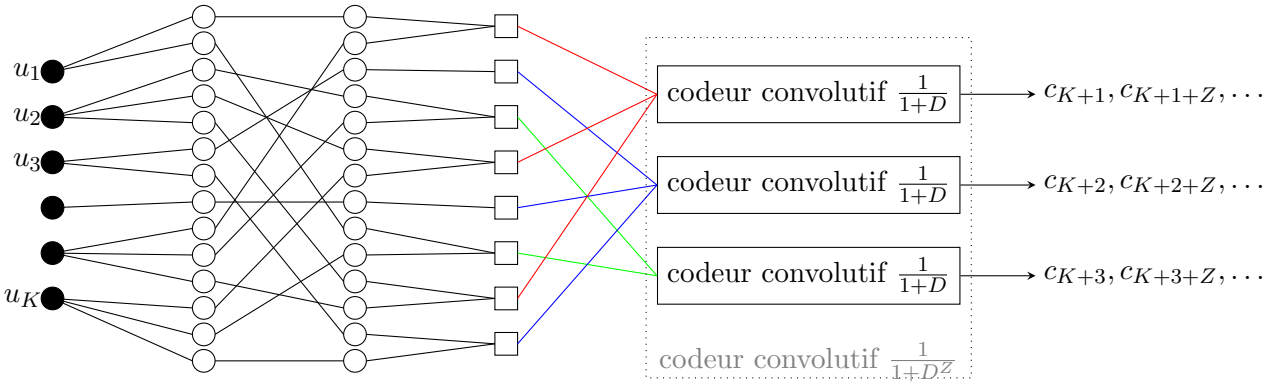


FIGURE 4.9 – Codage avec parallélisation d'un code LDPC avec une structure convolutive

Si la structure convolutive ne concerne pas tous les bits de parité (c'est-à-dire, que la double diagonale est présente uniquement sur les toutes dernières colonnes de $\boldsymbol{\mathcal{H}}$, et non pas sur les $N-K$ dernières), une étape est ajoutée au codage afin de déterminer la valeur des premiers bits de parité. Une fois ces bits de parité calculés, le codage convolutif peut être appliqué. Dans la section consacrée aux codes LDPC normés avec une structure convolutive, une méthode permettant de déterminer les valeurs de ces bits de parité sera expliquée si la structure de la matrice de parité le nécessite.

La structure convolutive de ces codes LDPC peut également être utilisée lors du décodage. En effet, un décodeur de type BCJR [1] peut être employé afin de corriger la partie convolutive. Ce décodage mêlant le décodage classique des codes LDPC avec le décodage des codes convolutifs permet de réduire le nombre d'itérations nécessaires.

4.3 Codage des codes LDPC quasi-cycliques

Le codage des codes LDPC quasi-cycliques s'effectue à partir de leur matrice génératrice qui est elle aussi quasi-cyclique.

Proposition 4.8. *La matrice génératrice d'un code LDPC quasi-cyclique est quasi-cyclique. Les sous-matrices cycliques des matrices génératrice et de parité sont de même taille, notée Z.*

Pour les codes LDPC avec une structure convolutive, la taille Z des sous-matrices cycliques définit l'écart entre les deux diagonales de la partie convolutive. D'où l'utilisation d'un codeur convolutif défini par $(1, \frac{1}{1+D^Z})$.

Les codes utilisés sont systématiques, la matrice génératrice $\boldsymbol{\mathcal{G}}$ est donc systématique, cependant le poids de ses $N-K$ dernières colonnes peut être élevé. Le codage par multiplication matricielle peut alors être coûteux. Une méthode de codage utilisant des registres à décalage est alors plus efficace et profite de la structure quasi-cyclique de la matrice génératrice. Ce codage est schématisé par la figure 4.10 (où $\otimes$ désigne la multiplication de chacun des éléments d'un vecteur par une même valeur binaire).

Le codage des codes LDPC quasi-cycliques peut donc faire appel à des registres à décalage. En effet, la matrice génératrice $\boldsymbol{\mathcal{G}}$ d'un code LDPC quasi-cyclique est de la forme $\boldsymbol{\mathcal{G}} = (Id|\boldsymbol{\mathcal{D}})$ où $\boldsymbol{\mathcal{D}}$ est une matrice quasi-cyclique de taille $K \times (N-K)$. Si $\boldsymbol{d}_i$ désigne la i-ème ligne de la matrice $\boldsymbol{\mathcal{D}}$, et que le mot d'information à coder est $\boldsymbol{u} = (u_1, \ldots, u_K)$ alors le mot de code correspondant est : $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{\mathcal{G}} = (u_1, \ldots, u_K, c_{K+1}, \ldots, c_N)$ avec $(c_{K+1}, \ldots, c_N) = u_1\boldsymbol{d}_1 \oplus u_2\boldsymbol{d}_2 \oplus \cdots \oplus u_K\boldsymbol{d}_K$. De plus, la quasi-cyclicité de $\boldsymbol{\mathcal{D}}$ implique une logique entre ses lignes, celle-ci est vérifiée sur des blocs de taille $Z \times Z$. Lors du codage, cette propriété est obtenue grâce aux registres à décalage qui sont réinitialisés tous les Z tops d'horloge. Au iZ-ème top d'horloge, l'initialisation est donnée par la ligne $\boldsymbol{d}_{iZ}$ (le premier registre est initialisé par les Z premiers bits de $\boldsymbol{d}_{iZ}$, le second par les Z bits suivants et ainsi de suite).
Au i-ème top d'horloge, la concaténation du contenu des registres à décalage permet de reconstruire la ligne $\boldsymbol{d}_i$ de la matrice $\boldsymbol{\mathcal{D}}$, celle-ci est ensuite multipliée par le i-ème bit d'information afin de déterminer $u_i\boldsymbol{d}_i$. Le vecteur ainsi calculé est ajouté (modulo 2) au vecteur obtenu précédemment, le vecteur résultat contient alors $u_1\boldsymbol{d}_1 \oplus u_2\boldsymbol{d}_2 \oplus \cdots \oplus u_i\boldsymbol{d}_i$. Au top d'horloge suivant, le contenu des registres est décalé d'un cran et le bit d'information u_{i+1}

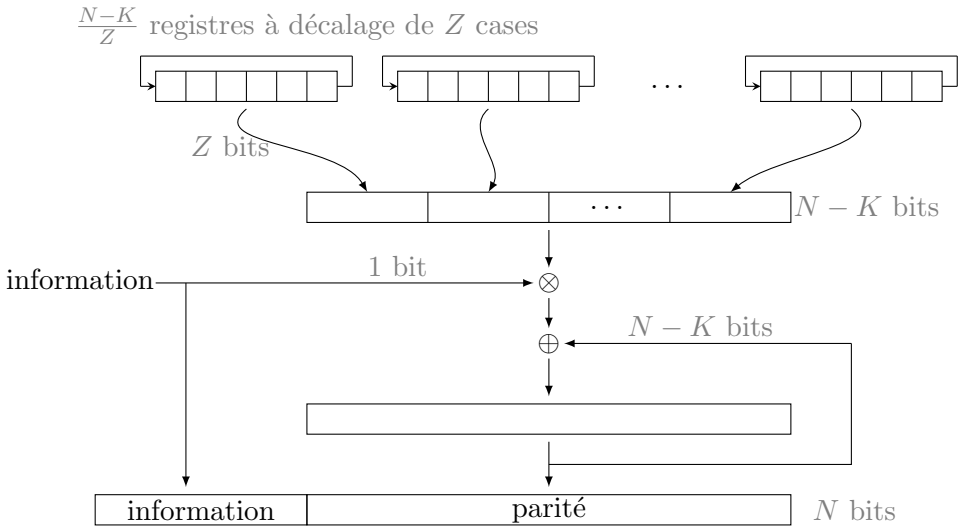


FIGURE 4.10 – Schéma de codage des codes LDPC quasi-cylciques

est entré dans le codeur. Lorsque tous les bits d'information ont été entrés dans le codeur, le vecteur résultat contient la parité recherchée. La concaténation de l'information et de la parité ainsi déterminée forme alors le mot de code.

Conclusion.
Ces méthodes de codages mettent en évidence l'intérêt d'utiliser des matrices structurées, le coût du codage en est ainsi diminué. De plus, le stockage des matrices génératrice et de parité est lui aussi beaucoup plus faible, en particulier pour les matrices quasi-cycliques. En effet, pour ces matrices, stocker une ligne sur Z est suffisant pour les déterminer entièrement. La densité des matrices de parité des codes LDPC assure un faible nombre de coefficients non nuls sur chacune des lignes, ainsi conserver uniquement les positions des coefficients non nuls d'une ligne sur Z est suffisant et ne nécessite que peu de place en mémoire.

Pour les codes LDPC qui sont à la fois quasi-cycliques et avec une structure convolutive (autrement dit S-IRA) les deux méthodes de codage décrites ci-dessus peuvent être utilisées.

D'autres méthodes de codage existent et utilisent plus finement les structures de la matrice de parité ou de la matrice génératrice. Les méthodes employées dépendent principalement de l'utilisation du code LDPC considéré.

Notations.
Les deux sections suivantes sont consacrées aux matrices de parité normées, afin de simplifier la représentation de ces matrices, les notations suivantes sont utilisées :

— Z : taille des sous-matrices cycliques de $\boldsymbol{\mathcal{H}}$ (par définition, Z divise N et $N-K$).
— P_i : matrice binaire de taille $Z \times Z$ obtenue par décalage cyclique de i positions vers la droite de la matrice identité. P_0 représente donc la matrice identité.

— (vide) : matrice nulle de taille $Z \times Z$ (elle est parfois notée $\mathbf{0}$ afin de faciliter la lecture).
— $\diamond$: matrice binaire de taille $Z \times Z$ nulle ou obtenue par décalage cyclique de la matrice identité.
— $\star$: matrice binaire de taille $Z \times Z$ cyclique (nulle ou obtenue en additionnant une ou plusieurs matrices identité décalées cycliquement vers la droite).

4.4 Codes LDPC avec une structure convolutive

Les codes LDPC ayant une matrice de parité avec une structure convolutive sont les plus représentés dans les normes de télécommunication. En effet, c'est le cas des codes LDPC utilisés dans les normes :

— ETSI DVB-C2 [45]
— ETSI DVB-S2 [42]
— ETSI DVB-S2X [41]
— ETSI DVB-T2 [43]
— ETSI DVB-NGH [44]
— ETSI DVB-RCS+M [40]
— ETSI GMR-1 [47]
— ETSI GMPRS-1 [46]
— IEEE 802.11 [61]
— IEEE 802.11n [65]
— IEEE 802.11ac [62]
— IEEE 802.11ad [63]
— IEEE 802.11af [64]
— IEEE 802.16 [67]
— IEEE 802.16e [68]
— IEEE 802.20 [69]
— IEEE 802.20.2 [60]
— IEEE 802.22 [70]
— ITU-T G9960 [112]
— WiMedia [141]
— CCSDS 131.5 [16]

Parmi ces codes, une majorité sont quasi-cycliques, ils sont présentés dans la sous-section suivante. D'autres peuvent être mis sous forme quasi-cyclique en effectuant une permutation sur les colonnes de la matrice de parité, ils font l'objet de la sous-section 4.4.2. Enfin, la sous-section 4.4.3 est consacrée aux codes LDPC normés qui possèdent une structure convolutive mais qui ne sont pas quasi-cycliques (même en effectuant des permutations).

Notations. Afin de limiter les répétitions, les normes IEEE 802.11, 802.11n, 802.11ac et 802.11af sont, à partir de maintenant, notées 802.11*. Les codes LDPC utilisés dans ces différentes normes sont identiques, il n'est donc pas utile de les distinguer. Remarquons que la norme 802.11ad n'est pas incluse dans la notation 802.11*, en effet, les codes LDPC utilisés dans cette norme sont différents. De la même façon, les normes IEEE 802.16 et 802.16e ainsi que 802.20 et 802.20.2 sont respectivement regroupées sous les notations 802.16* et 802.20*.

4.4.1 Codes LDPC quasi-cycliques avec une structure convolutive

Les codes LDPC normés possédant une structure convolutive et étant quasi-cycliques ne sont pas S-IRA au sens où la partie convolutive ne concerne pas les $N-K$ dernières colonnes de la matrice de parité. Cependant, leurs structures sont très proches des S-IRA mais varient légèrement d'une norme à une autre.

a. Normes IEEE 802.11*, 802.16*, 802.22 et WiMedia

Dans ces normes [61,62,64,65,67,68,70,141], les codes LDPC utilisés sont quasi-cycliques, avec des sous-matrices cycliques de taille $Z \times Z$ où

$$Z = \begin{cases} N/24 & \text{pour les normes IEEE 802.11*, 802.16* et 802.22} \\ 30 & \text{pour la norme WiMedia} \end{cases}$$

Schématiquement, toutes les matrices de parité de ces codes sont de la forme :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{ccccc|ccccccc} \diamond & \cdots & \cdots & \cdots & \diamond & P_i & P_0 & & & & \\ \diamond & \cdots & \cdots & \cdots & \diamond & & P_0 & P_0 & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & \ddots & \ddots & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & P_j & & & \ddots & \ddots & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & & & \ddots & P_0 \\ \diamond & \cdots & \cdots & \cdots & \diamond & P_i & & & & & P_0 \end{array}\right)$$

Cette structure, très proche de celle des codes LDPC dits S-IRA, leur a donné le nom de S-IRA Influenced. Les autres codes LDPC normés avec une structure convolutive et quasi-cycliques ont des structures dérivées de celle-ci, la dénomination S-IRA Influenced sera lors utilisée pour désigner cette structure de base.

Afin de mieux visualiser ces matrices de parités, l'une d'entre elles est donnée par la figure 4.11.

Dans les normes, ces codes sont irréguliers, de longueur N comprise entre 384 et 2 304 et de rendement $R = 1/2, 2/3, 3/4$ ou $5/6$. Le poids des lignes de la matrice de parité varie en fonction du rendement R du code, il est de :
— 7 et 8 pour $R = 1/2$,
— 11 pour $R = 2/3$,
— 14 et 15 pour $R = 3/4$,
— entre 19 et 22 pour $R = 5/6$.

Pour chaque norme et chaque rendement, le polynôme $\rho(x)$ définissant précisément le poids des lignes de la matrice $\boldsymbol{\mathcal{H}}$ est donné en annexe A.

Codage de ces codes LDPC. Une façon de coder ces codes LDPC consiste à utiliser leur structure convolutive. Celle-ci ne concerne pas les $N - K$ dernières colonnes de $\boldsymbol{\mathcal{H}}$ mais seulement les $N - K - Z$ dernières. La méthode de codage présentée dans la section précédente ne peut donc pas être directement appliquée, une étape supplémentaire est à effectuer. Celle-ci consiste à additionner par bloc les lignes de la matrice de parité. La structure de $\boldsymbol{\mathcal{H}}$ implique une structure sur les Z lignes ainsi obtenues. En effet, elles sont de la forme $(\star, \ldots, \star, P_j, 0, \ldots, 0)$ (voir figure 4.12). Les Z premiers bits de parité sont déduits à partir des équations de parité ainsi formées. Effectivement, grâce à cette matrice des dépendances impliquant uniquement des bits d'information et un unique bit de parité sont connues.
Une fois ces premiers bits de parité calculés, les suivants sont déterminés en utilisant des codeurs convolutifs comme expliqué précédemment. Pour cela, les Z premiers bits de parité jouent le même rôle que les bits d'information, ils sont donc dupliqués et permutés avant d'être additionnés puis entrés dans le codeur convolutif. (Les Z dernières lignes de $\boldsymbol{\mathcal{H}}$ sont ignorées lors de cette étape, les équations de parité définies par ces lignes seront, par construction, vérifiées par les mots de code ainsi obtenus.)

b. Norme WiMedia (deuxième structure)

Dans la norme WiMedia [141], deux longueurs de code LDPC sont utilisées. Pour la longueur la plus petite ($N = 1\,200$) la matrice de parité possède la structure présentée dans

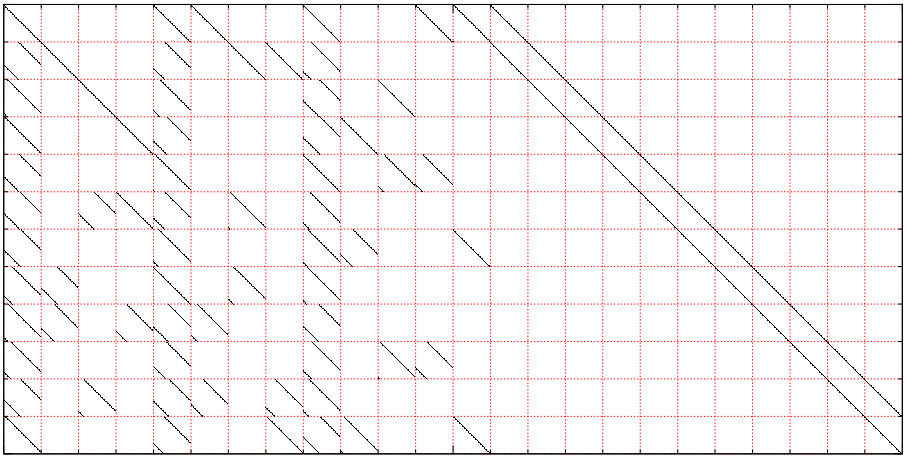


FIGURE 4.11 – Norme IEEE 802.11n, $R = 1/2$, $N = 1\,296$, $K = 648$ et $Z = 54$

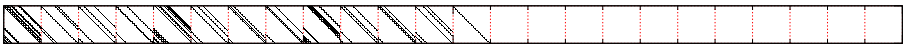


FIGURE 4.12 – Matrice obtenue en additionnant par bloc les lignes de la matrice de parité de la norme 802.11n pour $R = 1/2$ et $N = 1\,296$ (figure 4.11)

le paragraphe précédent (S-IRA Influenced), ce qui n'est pas le cas des codes LDPC de longueur $N = 1\,320$. Ces codes ont une matrice de parité obtenue par extension des matrices de parité des codes plus petits. La forme de ces matrices de parité peut être schématisée de la façon suivante :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{ccccc|cccccccccc} \diamond & \cdots & \cdots & \cdots & \diamond & P_i & P_0 & & & & & & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & P_0 & P_0 & & & & & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & \ddots & \ddots & & & & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & P_j & & & \ddots & \ddots & & & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & & & \ddots & P_0 & & & & \\ \diamond & \cdots & \cdots & \cdots & \diamond & P_i & & & & & P_0 & & & & \\ \diamond & \cdots & \cdots & \cdots & \diamond & \diamond & \cdots & \cdots & \cdots & \cdots & \diamond & P_s & P_0 & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \vdots & P_t & P_0 & P_0 & \\ \vdots & \cdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \vdots & & & P_0 & P_0 \\ \diamond & \cdots & \cdots & \cdots & \diamond & \diamond & \cdots & \cdots & \cdots & \cdots & \diamond & P_s & & & P_0 \end{array}\right)$$

Comme précédemment, il s'agit de codes irréguliers et les sous-matrices cycliques sont de taille $Z \times Z$ avec $Z = 30$. Avec cette longueur de code, les rendements formés sont $R = 5/11, 6/11, 15/22$ et $8/11$. Le poids des lignes de la matrice $\boldsymbol{\mathcal{H}}$ varie entre 5 et 15 en fonction du rendement considéré (voir en annexe A pour plus de précision).

Cette structure peut être vue comme la concaténation de deux matrices de parité de type S-IRA Influenced. Le codage de ces codes se découpe en deux étapes qui sont enchaînées,

chacune correspondant à un codage à partir d'une des deux sous-matrices qui compose la matrice de parité.

Codage de ces codes LDPC. La partie basse de la matrice de parité est toujours composée de 4 blocs de Z lignes, la partie haute possède donc $N-K-4Z$ lignes. En utilisant uniquement ces $N-K-4Z$ premières lignes, la méthode de codage définie pour les matrices de parité de type S-IRA Influenced est appliquée (addition des blocs afin de déterminer les Z premiers bits de parité, puis utilisation de la méthode de codage des codes LDPC avec une structure convolutive pour trouver les bits de parité suivants). Les $N-K-4K$ premiers bits de parité sont maintenant connus, la même méthode de codage est de nouveau appliquée mais en utilisant la matrice définie par les $4Z$ dernières lignes de $\boldsymbol{\mathcal{H}}$. Pour ce codage, les $N-K-4Z$ bits de parité déterminés précédemment jouent le rôle de bits d'information. En effet, cette seconde étape de codage, correspond à l'encodage par un code LDPC de longueur N et de dimension $N-4Z$.

c. Norme IUT-T G.9960

Dans la norme IUT-T G.9960 [112], les matrices de parité ont une structure très proche de la structure dite S-IRA Influenced. Elle peut être schématisée par :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{ccccc|ccccccc} \diamond & \cdots & \cdots & \cdots & \diamond & & P_0 & & & & \\ \diamond & \cdots & \cdots & \cdots & \diamond & P_i & P_0 & P_0 & & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & \ddots & \ddots & & \\ \vdots & \cdots & \cdots & \cdots & \vdots & P_j & & & \ddots & \ddots & \\ \vdots & \cdots & \cdots & \cdots & \vdots & & & & & \ddots & P_0 \\ \diamond & \cdots & \cdots & \cdots & \diamond & P_i & P_k & & & & P_0 \end{array}\right)$$

Les différences avec la structure S-IRA Influenced portent sur la position du premier bloc non nul de la partie de parité. En effet, celui-ci ne se trouve plus sur les Z premières lignes mais sur les Z suivantes. De plus, un bloc obtenu par décalage de la matrice identité est également ajouté sur les colonnes correspondant au second bloc de bits de parité. Cependant, comme dans les normes 802.11*, 802.16*, 802.22 la taille Z des sous-matrices cycliques est définie par $Z = N/24$.

Dans la norme IUT-T G9960, les codes utilisés sont irréguliers et de rendement $R = 1/2$, $2/3$ et $5/6$. Les lignes de la matrice $\boldsymbol{\mathcal{H}}$ sont de poids compris entre :
- 5 et 7 lorsque $R = 1/2$,
- 9 et 11 pour $R = 2/3$,
- 19 et 21 quand $R = 5/6$,

(plus de précision sur ces poids sont donnés en annexe A). Dans cette norme, sept longueurs de code sont normalisées, elles sont comprises entre 336 et 8 640.

Codage de ces codes LDPC. Le codage à partir de ces matrices de parité s'effectue en 3 étapes, les deux premières servant à déterminer les premiers bits de parité afin de pouvoir ensuite effectuer un codage utilisant la structure convolutive de ces codes.
D'après le schéma représentant ces matrices de parité, les Z premières lignes de $\boldsymbol{\mathcal{H}}$ définissent

Z équations de parité faisant intervenir des bits d'information et un unique bit de parité. Grâce à ces équations, les bits de parité concernés peuvent être calculés, il s'agit des bits c_{K+Z+1} à c_{K+2Z+1}. Ensuite, en additionnant par bloc les lignes de la matrice $\boldsymbol{\mathcal{H}}$, Z équations de parité sont déduites. Par construction, ces nouvelles équations de parité font intervenir un (et un seul) des Z premiers bits de parité ainsi qu'un des bits de parité déterminé lors de la première étape du codage. Le bit de parité dont la valeur n'est pour l'instant pas connue est donc défini par un ensemble de bits connus, il en est alors déduit.
Une fois ces deux étapes effectuées, les $2Z$ premiers bits de parité sont connus, ils peuvent alors être utilisés de la même façon que des bits d'information pour le codage des derniers bits de parité. Ce codage s'effectue en utilisant la structure convolutive de $\boldsymbol{\mathcal{H}}$ et la méthode décrite au début de ce chapitre.

d. Normes IEEE 802.20*

Une dernière variante de la structure S-IRA Influenced est celle présente dans les normes IEEE 802.20* [60, 69]. Dans ces normes, les matrices de parité sont de la forme suivante :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{ccccc|ccccccccccc}
\diamond & \cdots & \cdots & \cdots & \diamond & P_i & P_0 & & & & & & & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & & P_0 & P_0 & & & & & & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & & & \ddots & \ddots & & & & & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & P_j & & & \ddots & \ddots & & & & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & & & & & \ddots & P_0 & & & & \\
\diamond & \cdots & \cdots & \cdots & \diamond & P_i & & & & & P_0 & & & & \\
\diamond & \cdots & \cdots & \cdots & \diamond & \diamond & \cdots & \cdots & \cdots & \cdots & \diamond & P_0 & & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \vdots & & P_0 & & \\
\vdots & \cdots & \cdots & \cdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \vdots & & & \ddots & \\
\diamond & \cdots & \cdots & \cdots & \diamond & \diamond & \cdots & \cdots & \cdots & \cdots & \diamond & & & & P_0
\end{array}\right)$$

Ces codes correspondent donc à la concaténation de deux codes LDPC, dont le premier est défini par une matrice de parité de type S-IRA Influenced.

Ces codes sont construits à partir de protographes et portent le nom de AR4JA (Accumulate Repeat 4 Jagged Accumulate) [36, 37, 126]. Ici, nous nous intéressons uniquement à la structure de la matrice de parité et non à la façon dont elle est obtenue.

Les codes LDPC utilisés dans cette norme sont donc irréguliers. La valeur de Z est variable, ce qui implique des codes de longueur variable (comprise entre $33Z$ et $58Z$). Les six rendements définis sont entre 2/11 et 11/58. Quel que soit le rendement et la longueur du code considéré, les lignes de la matrice de parité $\boldsymbol{\mathcal{H}}$ sont de poids, au plus, 7.

4.4.2 Codes LDPC avec une structure convolutive presque quasi-cycliques après permutation

Les normes ETSI utilisant des codes LDPC [40–47] ainsi que la norme CCSDS 131.5 [16], font appel à des codes avec une structure convolutive mais qui sont, à première vue, non

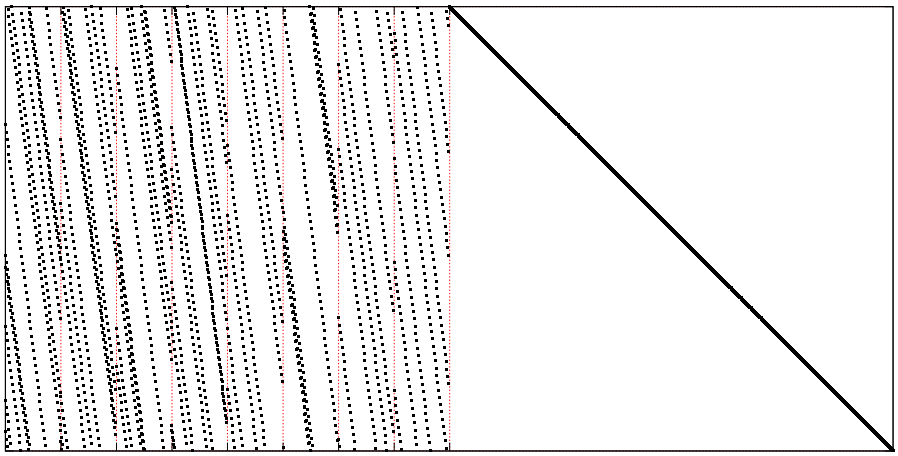


FIGURE 4.13 – Norme ETSI GMR-1, $N = 976$, $K = 488$, $R = 1/2$ et $Z = 61$.

quasi-cycliques. Cependant, leur construction est extrêmement structurée et en appliquant une permutation (elle aussi très structurée) sur les colonnes des matrices de parité normées, il est possible d'obtenir des matrices presque quasi-cycliques (au sens où elle ne diffère que dans une position d'une matrice quasi-cyclique) mais ne possédant plus de structure convolutive.

La construction des matrices de parité des codes LDPC utilisés dans les normes ETSI DVB-(C2, S2, S2X, T2, NGH, RCS+M), GMR-1, GMPRS-1 et CCSDS 131.5 est la suivante : sur la partie de redondance de $\boldsymbol{\mathcal{H}}$ (c'est-à-dire les $N - K$ dernières colonnes) seuls les éléments de la diagonale et de la sous-diagonale sont non-nuls. $\boldsymbol{\mathcal{H}}$ possède donc une structure convolutive et peut être schématisée comme suit :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{c|ccccc} & 1 & & & & \\ & 1 & 1 & & & \\ \boldsymbol{\mathcal{A}} & & 1 & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & 1 & 1 \end{array}\right)$$

Quant-à la matrice $\boldsymbol{\mathcal{A}}$, elle est construite par blocs de Z colonnes. Dans un bloc, la i-ème colonne est égale à la $(i-1)$-ème décalée cycliquement de q positions vers le bas (avec $q = (N-K)/Z$). Ces matrices sont irrégulières, il s'agit donc de codes LDPC dits IRA. Les longueurs des codes normés de ce type sont très variables, entre 950 et 64 800 pour les normes ETSI et entre 576 et 24 576 pour la norme CCSDS 131.5. Le poids des lignes de $\boldsymbol{\mathcal{H}}$ varie lui aussi, il est au minimum de 3 (pour les faibles rendements) et au maximum de 37 (pour les forts rendements) des normes ETSI. Pour la norme CCSDS 131.5, le poids les lignes de la matrice de parité est compris entre 10 et 55. Toutes les longueurs et dimensions normées, ainsi que les polynômes indiquant la répartition de poids sont donnés en annexe A.

Afin de mieux visualiser la structure de ces codes LDPC, la figure 4.13 représente une des matrices normées de cette forme.

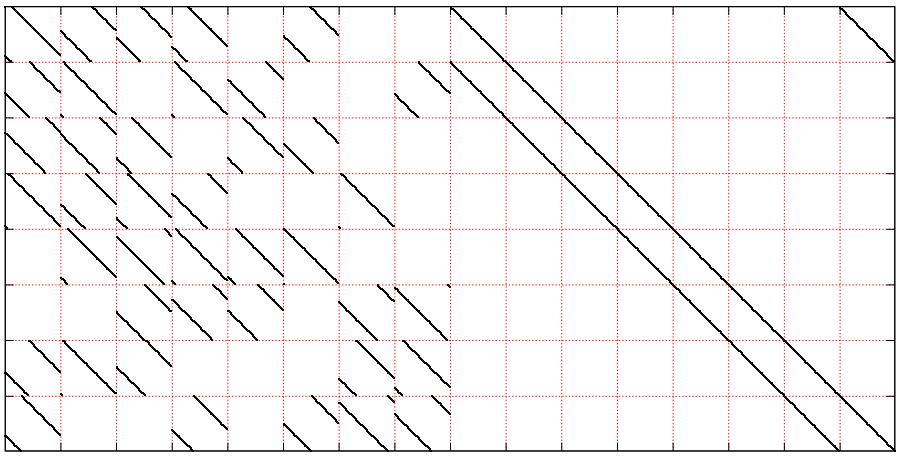


FIGURE 4.14 – Norme ETSI GMR-1, $N = 976$, $K = 488$, $R = 1/2$ et $Z = 61$ après permutation des lignes et des colonnes.

En permutant les lignes puis les colonnes de $\boldsymbol{\mathcal{H}}$, la matrice $\boldsymbol{\mathcal{H}}'$ obtenue est une matrice de parité d'un code LDPC équivalent au code d'origine. Elle peut être représentée de la façon suivante. Remarquons que la permutation sur les colonnes concerne uniquement les colonnes de la partie de parité.

$$\boldsymbol{\mathcal{H}}' = \left(\begin{array}{cccc|cccccc} \star & \dots & \dots & \star & P_0 & & & & & \widetilde{P_{-1}} \\ \vdots & \dots & \dots & \vdots & P_0 & P_0 & & & & \\ \vdots & \dots & \dots & \vdots & & \ddots & \ddots & & & \\ \vdots & \dots & \dots & \vdots & & & \ddots & \ddots & & \\ \vdots & \dots & \dots & \vdots & & & & \ddots & \ddots & \\ \star & \dots & \dots & \star & & & & & P_0 & P_0 \end{array}\right)$$

avec $\widetilde{P_{-1}}$ définie par

$$\widetilde{P_{-1}} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \ddots & & & \vdots \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

La sous-matrice $\widetilde{P_{-1}}$ n'est pas quasi-cyclique (à cause de son coefficient nul dans le coin supérieur droit). La matrice $\boldsymbol{\mathcal{H}}'$ n'est donc pas quasi-cyclique, mais presque quasi-cyclique. C'est également à cause de ce coefficient nul dans $\widetilde{P_{-1}}$ que $\boldsymbol{\mathcal{H}}'$ ne possède pas de structure convolutive. La matrice $\boldsymbol{\mathcal{H}}'$ est donc presque quasi-cyclique et ses sous-matrices sont de taille $Z \times Z$. Dans la majorité des normes utilisant des codes de ce type, Z est égal à 360 mais il varie entre 8 et 128 pour les codes de petite longueur ($\leq 32\,000$).

Pour illustration, après permutation des lignes et des colonnes la matrice de parité de la figure 4.13 devient la matrice de la figure 4.14.

4.4.3 Codes LDPC non quasi-cycliques avec une structure convolutive

La norme CCSDS 131.5 [16] est l'unique norme qui utilise des codes LDPC avec une structure convolutive mais non quasi-cycliques. Dans cette norme, seule la construction de la matrice de parité du code utilisée est définie ; il s'agit de codes LDPC dits flexibles et notés F-LDPC [89, 110]. La matrice de parité de ces codes est de la forme :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{c|c|c} \boldsymbol{\mathcal{H}}_o & Id & \mathbf{0} \\ \hline \multicolumn{2}{c|}{\boldsymbol{\mathcal{H}}_u} & \boldsymbol{\mathcal{H}}_p \end{array}\right)$$

où $\boldsymbol{\mathcal{H}}_0$ est une petite matrice binaire non-structurée, chacun de ses coefficients est choisi aléatoirement (ils ont la même probabilité d'être nul que d'être non-nul). La matrice $\boldsymbol{\mathcal{H}}_u$ est définie par le poids de ses colonnes et elle est construite à partir d'une permutation aléatoire. Quant à la sous-matrice $\boldsymbol{\mathcal{H}}_p$ c'est elle qui assure la partie convolutive du code ainsi défini, elle est composée de la diagonale et de la sous-diagonale :

$$\boldsymbol{\mathcal{H}}_p = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & \ddots & & \vdots \\ 0 & 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & 1 \end{pmatrix}$$

Les codes LDPC possédant de telles matrices de parité sont particulièrement adaptés au cas du canal à effacement. Dans la norme CCSDS 131.5 les longueurs et rendements des codes utilisés ne sont pas explicités, seuls deux exemples sont cités afin d'illustrer la performance de ces codes, ils sont de longueurs 1 536 et 12 288 et de rendement 2/3.

4.5 Codes LDPC sans structure convolutive

Certaines normes utilisent des codes LDPC qui ne possèdent pas de structure convolutive, ce sont les formes très variées des matrices de parité de ces codes qui vont être listées dans cette section. Les normes utilisant de tels codes LDPC sont :

- IEEE 802.11ad [63]
- IEEE 802.15.3c [66]
- IEEE 802.3 [71]
- IEEE 802.3an [72]
- IEEE 1901 [59]
- CCSDS 131.1 [15]
- CCSDS 231.1 [17]
- CMMB [26]

Comme pour les matrices de parité avec une structure convolutive, ces matrices peuvent être classées en plusieurs groupes en fonction de leur quasi-cyclicité. En effet, elles peuvent être quasi-cycliques par construction, ou être mises sous forme quasi-cyclique par permutation des lignes et des colonnes ou ni l'un ni l'autre, dans ce cas, elles sont dites non quasi-cycliques.

4.5.1 Codes LDPC quasi-cycliques sans structure convolutive

a. Normes IEEE 802.11ad et 802.15.3c

Les codes LDPC utilisés dans la norme IEEE 802.11ad ainsi qu'une partie de ceux de la norme IEEE 802.15.3c possèdent une matrice de parité qui peut être schématisée de la façon suivante :

$$\boldsymbol{\mathcal{H}} = \left(\begin{array}{ccccc|ccccc} \diamond & \cdots & \cdots & \cdots & \diamond & P_0 & & & & \\ \vdots & \cdots & \cdots & \cdots & \cdots & \ddots & P_0 & & & \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & & \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & \\ \diamond & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \diamond & P_0 \end{array}\right)$$

Même si la matrice de parité, explicitement donnée dans la norme, n'est pas sous cette forme, elle peut y être ramenée par une simple permutation de ses lignes.
La figure 4.15 représente la matrice de parité d'un des codes LDPC normés de ce type.

Dans les normes IEEE 802.11ad et 802.15.3c, ces codes LDPC sont irréguliers, de longueur $N = 672$, les sous-matrices cycliques sont de taille $Z \times Z$ avec $Z = 42$ dans la norme 802.11ad et $Z = 21$ dans la norme 802.15.3c et leurs rendements varient entre $1/2$ et $7/8$. Comme pour les autres structures, tous les paramètres normés sont listés en annexe A avec les polynômes définissant le poids des lignes de la matrice de parité.

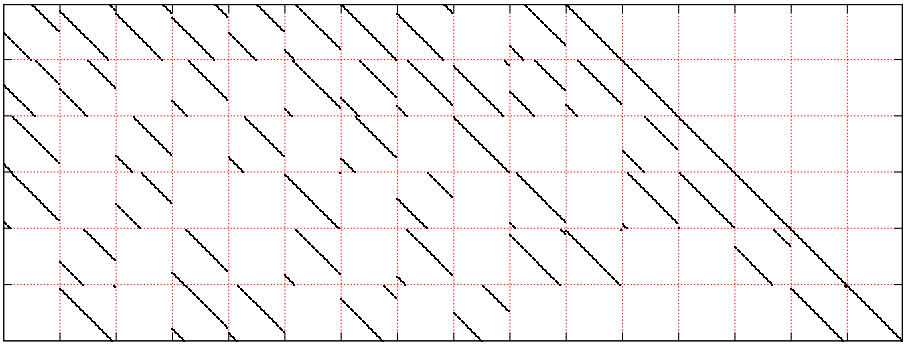


FIGURE 4.15 – Norme IEEE 802.11ad, $N = 672$, $K = 420$, $R = 5/8$ et $Z = 42$

b. Norme IEEE 802.15.3c (deuxième structure)

La norme IEEE 802.15.3c utilise un autre code LDPC, ce code possède une matrice de parité quasi-cyclique mais celle-ci ne peut pas être mise sous forme triangulaire inférieure contrairement à la structure précédente. La représentation schématique de cette matrice est donc

$$\boldsymbol{\mathcal{H}} = \begin{pmatrix} \diamond & \cdots & \diamond \\ \vdots & \ldots & \vdots \\ \diamond & \cdots & \diamond \end{pmatrix}$$

Cette matrice de parité (représentée sur la figure 4.16) est irrégulière et définit un code LDPC de longueur $N = 672$, de dimension $K = 336$ et donc de rendement $R = 1/2$. Ses sous-matrices cycliques sont de taille $Z \times Z$ avec $Z = 21$ et ses lignes sont de poids 6, 7 et 8.

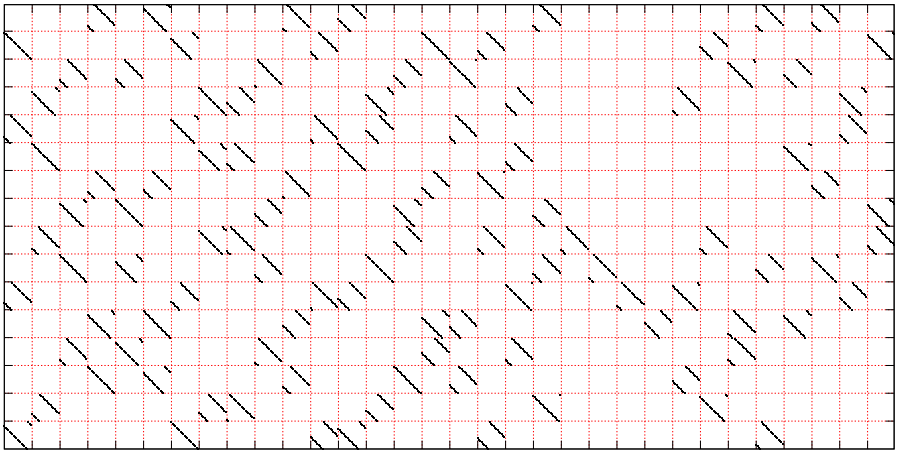


FIGURE 4.16 – Norme IEEE 802.15.3c $N = 672$, $K = 336$, $R = 1/2$ et $Z = 21$

c. Norme CCSDS 131.1 (première structure)

Deux types de codes LDPC sont utilisés dans la norme CCSDS 131.1 [15], le premier d'entre-eux est obtenu à partir de la géométrie euclidienne, d'où son nom LDPC QC-EG (Quasi-Cyclic Euclidean Geometry) [78]. Seule la structure de la matrice de parité est considérée, nous ne nous intéresserons pas plus précisément à sa construction. Schématiquement ce type de codes LDPC peut être représenté par :

$$\boldsymbol{\mathcal{H}} = \begin{pmatrix} \star & \cdots & \cdots & \star \\ \star & \cdots & \cdots & \star \end{pmatrix}$$

Plus précisément, le code LDPC de cette forme, utilisé dans la norme CCSDS 131.1, possède une matrice de parité composée de 2×16 sous-matrices cycliques de taille $Z \times Z$ avec $Z = 511$. Chaque sous-matrice est obtenue en additionnant deux matrices identité décalées cycliquement vers la droite. Cette matrice est représentée sur la figure 4.17.

Cette matrice de parité est régulière, toutes ses lignes sont de poids 32 et toutes ces colonnes de poids 4. Elle définie un code de longueur $N = 8\,176$ et de dimension $K = 7\,156$. Il peut être remarqué que cette matrice de parité n'est pas de rang plein, en effet, elle possède $1\,022$ lignes alors que $N - K = 1\,020$ (elle contient donc deux sommes de lignes qui sont nulles).

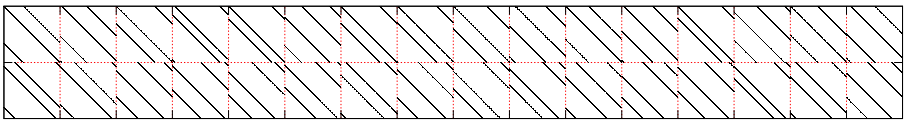


FIGURE 4.17 – Norme CCSDS 131.1, $N = 8\,176$, $K = 7\,156$ et $Z = 511$

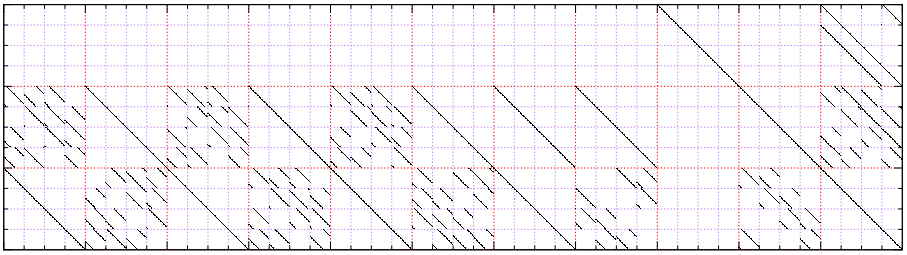


FIGURE 4.18 – Norme CCSDS 131.1, $N = 1\,408$, $K = 1\,024$ et $Z = 32$

d. Norme CCSDS 131.1 (deuxième structure)

D'autres codes LDPC sont utilisés dans la norme CCSDS 131.1 [15], ils sont construits à partir de protographes [36, 37, 126]. Ils s'agit de codes quasi-cycliques, la méthode de codage des codes LDPC quasi-cycliques présentée précédemment est décrite dans cette norme.

Dans la norme CCSDS 131.1, les codes de ce type sont irréguliers, leurs sous-matrices cycliques sont de taille $Z \times Z$ avec Z valant 128, 256, 512, 1 024, 2 048, 4 096, ou 8 192. Les différents paramètres normés sont listés en annexe A.

Sur la figure 4.18 est représenté le code de plus haut rendement ($R = 4/5$) utilisé dans cette norme. Les matrices de parité des codes de rendement inférieur sont des sous-matrices de celle-ci. Par exemple, pour obtenir la matrice de parité du code de rendement $R = 3/4$ il suffit de tronquer les $8Z$ premières colonnes de la matrice de parité définissant le code de rendement $R = 4/5$.

Il peut être noté que, dans cette norme, les $4Z$ derniers bits de redondance sont toujours poinçonnés lorsqu'un code LDPC de ce type est utilisé.

e. Norme CCSDS 231.1

La norme CCSDS 231.1 [17] fait appel à des codes LDPC binaires mais également à des codes LDPC non binaires. Trois codes LDPC binaires sont normés, ils sont de petite taille (128, 256 et 512) et de rendement 1/2. Leurs matrices de parité sont de la forme :

$$\boldsymbol{\mathcal{H}} = \begin{pmatrix} \star & \diamond & \diamond & \diamond & \mathbf{0} & \diamond & \diamond & P_0 \\ \diamond & \star & \diamond & \diamond & P_0 & \mathbf{0} & \diamond & \diamond \\ \diamond & \diamond & \star & \diamond & \diamond & P_0 & \mathbf{0} & \diamond \\ \diamond & \diamond & \diamond & \star & \diamond & \diamond & P_0 & \mathbf{0} \end{pmatrix}$$

Afin de mieux visualiser cette structure, la figure 4.19 représente la plus grande des matrices normalisées. Toutes les lignes des matrices normées sont de poids 8, en effet, dans la structure ci-dessus, les sous-matrices notées par $\star$, sont toutes obtenues en additionnant deux matrices identités décalées cycliquement vers la droite.

Quant aux codes LDPC non-binaires utilisés dans cette norme ils sont également très structurés et définis sur $GF(256)$ et $GF(16)$.

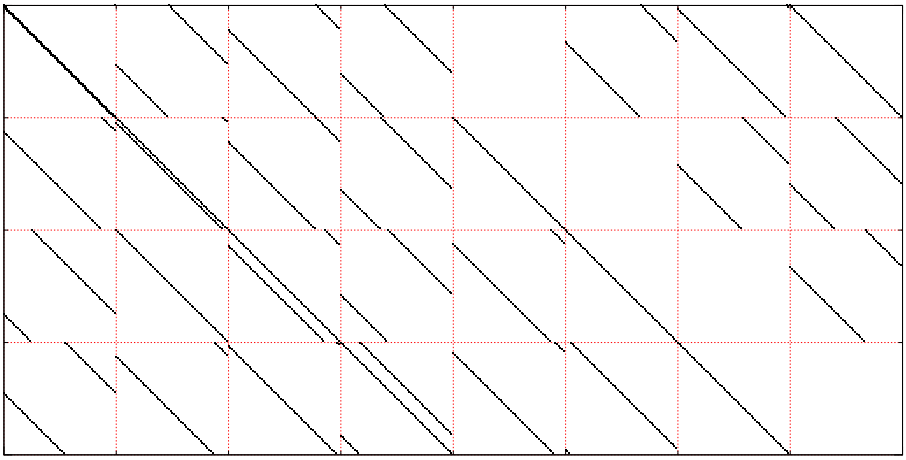


FIGURE 4.19 – Norme CCSDS 231.1, $N = 512$, $K = 256$ et $Z = 64$

4.5.2 Codes LDPC sans structure convolutive quasi-cycliques après permutation

Certains codes LDPC normés, tels que ceux utilisés dans les normes CMMB [26] et IEEE 802.15.3c [66], ne possèdent pas de structure convolutive et sont, à première vue, non quasi-cycliques. Cependant, en appliquant une permutation (très structurée) sur les colonnes des matrices de parités normées, une matrice de parité quasi-cyclique d'un code équivalent peut être obtenue.

a. Norme CMMB

Deux codes LDPC sont définis dans cette norme [26], ils sont de rendements $R = 1/2$ et $R = 3/4$. Quel que soit le rendement, la matrice de parité $\boldsymbol{\mathcal{H}}$ peut être représentée de la façon suivante :

$$\boldsymbol{\mathcal{H}} = \begin{pmatrix} H_1 & H_{256} & H_{255} & \cdots & \cdots & H_2 \\ H_2 & H_1 & H_{256} & \ddots & & H_3 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & H_{255} \\ \vdots & & & \ddots & \ddots & H_{256} \\ H_{256} & H_{255} & \cdots & \cdots & H_2 & H_1 \end{pmatrix}$$

où H_i désigne une matrice binaire de taille $Z_1 \times Z_2$ avec
— pour $R = 1/2$: $Z_1 = 18$ et $Z_2 = 36$
— pour $R = 3/4$: $Z_1 = 9$ et $Z_2 = 36$

Dans la norme CMMB, ces codes sont réguliers et de longueur $N = 9\,216$. Les lignes des matrices de parité sont de poids 6 ou 12 selon le rendement considéré. Pour obtenir de tels poids, la plupart des sous-matrices H_i sont nulles.

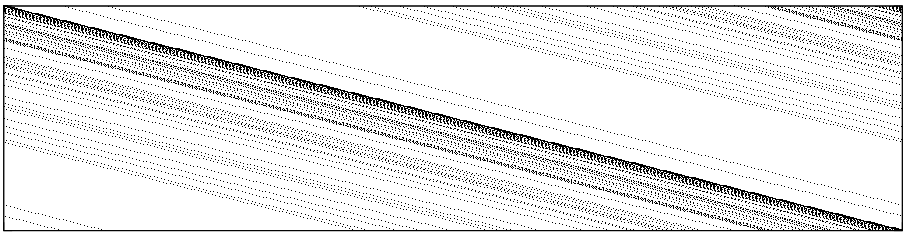


FIGURE 4.20 – Norme CMMB, $N = 9\,216$, $K = 6\,912$ et $R = 3/4$

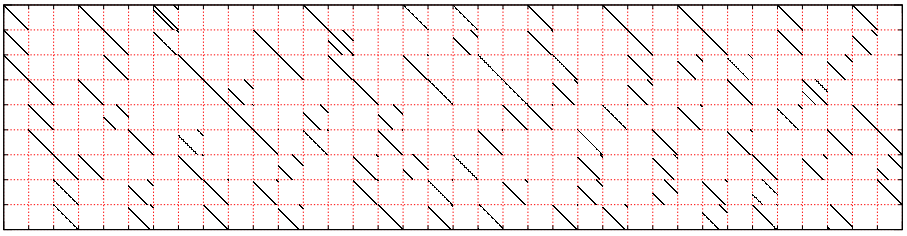


FIGURE 4.21 – Norme CMMB, $N = 9\,216$, $K = 6\,912$ et $R = 3/4$ après permutation des lignes et des colonnes

La figure 4.20 représente une de ces matrices de parité, telle que donnée dans la norme. Proportionnellement aux dimensions de la matrice, les sous-matrices H_i sont petites et ne sont par conséquent pas très visibles sur la figure. Quant à la figure 4.21, elle représente la matrice obtenue après permutation des lignes et des colonnes.

b. Norme IEEE 802.15.3c (troisième structure)

Un troisième type de code LDPC est utilisé dans la norme IEEE 802.15.3c [66], celui-ci définit des codes réguliers de rendement $R = 14/15$. La seule longueur pour laquelle il est normé est $N = 1\,440$.

Cette matrice de parité est composée d'une juxtaposition de sous-matrices binaires rectangles :

$$\boldsymbol{\mathcal{H}} = \Big(H_1, H_2, \ldots, H_{96}\Big)$$

où H_{i+1} est de taille 96×15 et est égale à H_i décalée cycliquement d'une unité vers le haut. Avec cette construction, toutes les lignes de $\boldsymbol{\mathcal{H}}$ sont du même poids. Dans la norme 802.15.3c, ce poids est de 45.

La structure très forte de cette matrice de parité permet de la mettre sous forme quasi-cyclique en appliquant une permutation sur ses lignes et ses colonnes. Les figures 4.22 et 4.23 illustrent la matrice de parité avant et après permutation.

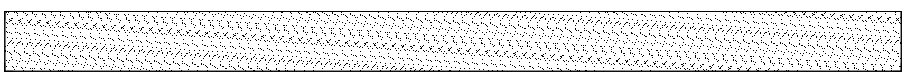


FIGURE 4.22 – Norme IEEE 802.15.3c $N = 1\,440$, $K = 1\,344$ et $R = 14/15$

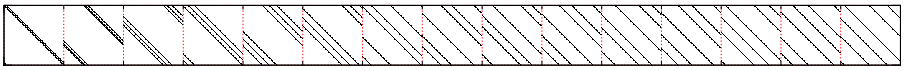


FIGURE 4.23 – Norme IEEE 802.15.3c $N = 1\,440$, $K = 1\,344$, $R = 14/15$, $Z = 96$ après permutation des lignes et des colonnes

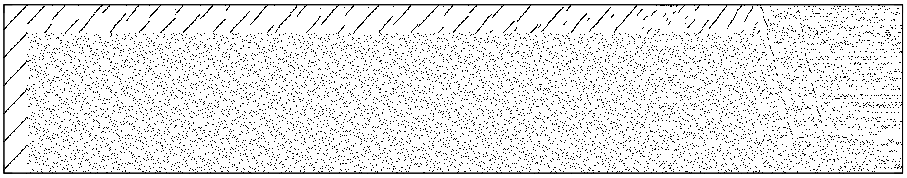


FIGURE 4.24 – Norme IEEE 802.3, $N = 2\,048$ et $K = 1\,723$

4.5.3 Codes LDPC sans structure convolutive non quasi-cycliques

Deux normes utilisent des codes LDPC qui ne possèdent pas de structure convolutive et qui sont non quasi-cycliques. Pour chacune d'elle la construction de la matrice de parité est très particulière.

a. Normes IEEE 802.3 et IEEE 802.3an

Un seul code LDPC est utilisé dans les normes IEEE 802.3 et 802.3an [71,72], il s'agit d'un code de longueur $N = 2\,048$ et de dimension $K = 1\,723$. La matrice de parité de ce code est construite à partir d'un code de Reed-Solomon qui a été fortement raccourci, seulement deux symboles d'information sont conservés [38]. La matrice de parité obtenue est régulière ; dans la norme, toutes ses lignes sont de poids 32 et toutes ses colonnes de poids 6. Cependant, cette matrice n'est pas de rang plein. Par exemple, la matrice explicitée dans cette norme possède $N - K + 59$ lignes.

Par construction, en appliquant une permutation sur les lignes et les colonnes de la matrice de parité, il est possible d'obtenir une matrice d'un code équivalent tel que celle-ci soit partitionnable en sous-matrices carrées où chacune d'elle est une matrice de permutation.

Dans les normes 802.3 et 802.3an, la matrice de parité $\boldsymbol{\mathcal{H}}$ est explicitée (représentée sur la figure 4.24), tout comme la matrice génératrice ainsi que les permutations à effectuer sur les lignes et les colonnes de $\boldsymbol{\mathcal{H}}$ de sorte à obtenir une matrice triangulaire inférieure.

b. Norme IEEE 1901

Dans la norme IEEE 1901 [59], les codes LDPC utilisés sont tout autant des codes convolutifs que des codes LDPC. En effet, ils possèdent une matrice de parité infinie qui est construite à partir de l'écriture binaire de trois polynômes. Ces trois polynômes sont répétés alternativement en les décalant vers la droite. Ces codes ont d'ailleurs étés présentés comme des codes convolutifs avec des matrices de parité creuses [49].

Dans la norme, les trois polynômes utilisés possèdent le même nombre de coefficients non-nuls. Ce poids varie entre 6 et 15 en fonction du rendement considéré. Sur la figure 4.25 est représentée la matrice de parité tronquée pour $R = 1/2$ et $N = 2\,000$.

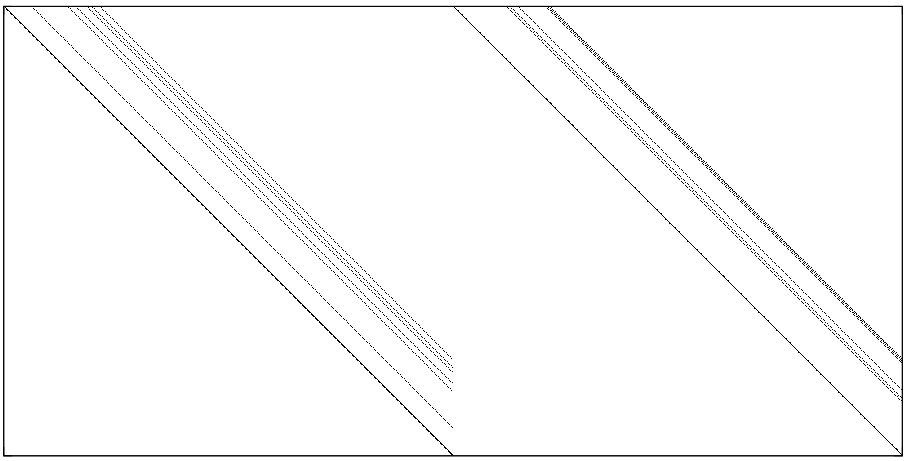


FIGURE 4.25 – Norme IEEE 1901, $R = 1/2$ et $N = 2\,000$

Partie II

Reconnaissance de codes correcteurs d'erreurs

Chapitre 5

Reconnaissance des turbo-codes

5.1 La reconnaissance des turbo-codes

Pour reconstituer un turbo-code, il nous faut non seulement reconstituer les deux codes convolutifs, mais aussi la permutation qui est utilisée. C'est ce dernier point qui se révèle très souvent être le plus délicat à effectuer.

La reconnaissance des turbo-codes a fait, ces dernières années, l'objet de diverses publications. La majorité de ces publications concernent la reconnaissance d'un ou des deux codes convolutifs des turbo-codes parallèles. C'est par exemple le cas des articles [33, 34, 150], c'est également le cas de [58, 100] mais uniquement sur des données non bruitées. L'article [108] décrit une méthode permettant de retrouver à partir de données bruitées quelques paramètres du turbo-code tels que le rendement des codes convolutifs et la longueur de la permutation. Quant à la méthode proposée dans l'article [125], elle a pour but de retrouver le motif du poinçonnage utilisé.
Enfin, les premiers articles à reconstruire la permutation des turbo-codes parallèles sont [25] et [29]. Avant eux, Barbier a proposé dans [2] une méthode pour reconstruire les turbo-codes, et en particulier la permutation interne, mais il s'est avéré que le problème résolu ne correspond pas au problème de la reconnaissance d'un turbo-code.

Un seul article [86] se consacre à la reconnaissance des turbo-codes série. Dans cet article les codes convolutifs sont reconnus, la permutation est également reconstruite mais ceci grâce à une forte hypothèse sur la structure de la permutation qui n'est donc pas aléatoire. En effet, pour cette méthode, la position des bits d'information est supposée connue.

Ce chapitre présente un nouvel algorithme de reconstruction de la permutation. Celui-ci est applicable aux cas des turbo-codes parallèles, des turbo-codes parallèles poinçonnés ainsi qu'aux turbo-codes série. Il permet de retrouver la permutation utilisée à partir d'un ensemble de mots de code bruités observés en sortie d'un canal de communication. Cette méthode fait l'objet d'une publication à la conférence ISIT 2014 [127] et améliore les méthodes existantes. Elle est d'abord présentée pour les turbo-code parallèles non poinçonnés, elle est ensuite adaptée afin de reconstruire la permutation des turbo-codes poinçonnés. De façon très similaire, elle est enfin appliquée aux turbo-codes série.

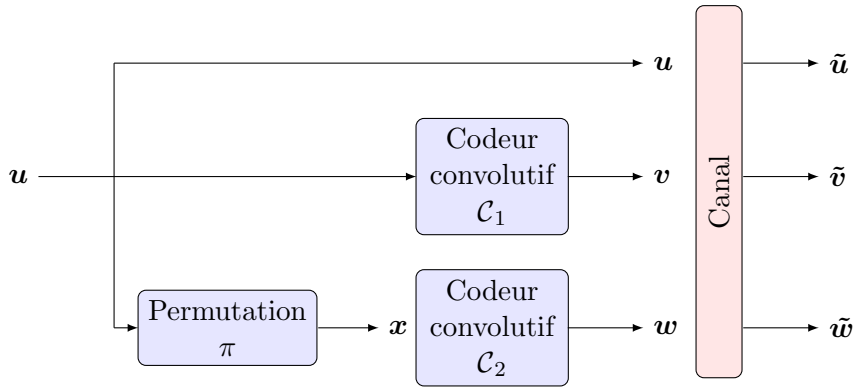


FIGURE 5.1 – Notations pour les turbo-codes parallèles

5.2 Cas des turbo-codes parallèles (non poinçonnés)

5.2.1 Notations et hypothèses

La liste des hypothèses nécessaires pour cette méthode est la suivante, cette liste est quasiment la même pour les trois types de turbo-codes (parallèles, parallèles poinçonnés et série). Les notations utilisées pour les turbo-codes parallèles sont regroupées sur la figure 5.1.

1. Le codeur convolutif $\mathcal{C}_2$ est connu, il est systématique et de type $(2, 1)$.
2. Le premier bit de redondance émis par le codeur $\mathcal{C}_2$ dépend du premier bit entrant dans ce codeur.
3. Les registres du codeur $\mathcal{C}_2$ sont initialisés à 0 au début du codage de chaque mot.
4. La permutation π est de longueur N.
5. La permutation π est choisie de manière uniforme parmi les $N!$ possibles.
6. Le canal utilisé est sans mémoire, son type ainsi que sa probabilité d'erreur sont connus.
7. M mots d'information $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^M$ sont choisis aléatoirement, indépendamment les uns des autres et suivant une distribution uniforme.
8. L'ensemble des M mots de code bruités $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ est donné.
9. Les trois sorties du codeurs sont séparées, autrement dit la décomposition des mots bruités en trois vecteurs est donnée : $\tilde{\boldsymbol{m}}^s = (\tilde{\boldsymbol{u}}^s, \tilde{\boldsymbol{v}}^s, \tilde{\boldsymbol{w}}^s)$, $\forall s \in \{1, \ldots, M\}$.

Le fait de connaître le type du canal ainsi que sa probabilité d'erreur permet de déterminer les probabilités $\boldsymbol{P}(u_t^s = 0|\tilde{u}_t^s)$, $\boldsymbol{P}(v_t^s = 0|\tilde{v}_t^s)$ et $\boldsymbol{P}(w_t^s = 0|\tilde{w}_t^s)$ pour tout $t \in \{1, \ldots, N\}$ et tout $s \in \{1, \ldots, M\}$. Pour les trois canaux binaires qui sont utilisés par la suite, ces calculs de probabilités peuvent s'effectuer en utilisant les rappels faits dans le chapitre 1.

Notations. Tous les vecteurs sont donc de taille N, leurs coefficients sont numérotés de 1 à N, par exemple $\boldsymbol{u} = (u_1, \ldots, u_N)$.
Si $\boldsymbol{a} = (a_1, a_2, \ldots, a_N)$ est un vecteur alors la notation $\boldsymbol{a}_{i..j}$ représente le sous-vecteur $(a_i, \ldots, a_j)$.

L'hypothèse la plus importante porte sur la connaissance du codeur convolutif $\mathcal{C}_2$. Dans la plupart des cas, cette hypothèse est vérifiée puisque les deux codeurs convolutifs sont généralement égaux et en utilisant les méthodes existantes de reconnaissance des codes convolutifs [5, 9, 23, 28, 35, 95, 101, 103, 104, 131, 134, 156] à partir des vecteurs $\tilde{\boldsymbol{u}}$ et $\tilde{\boldsymbol{v}}$ le code $\mathcal{C}_1$ est

retrouvé. De plus, si les deux codes convolutifs sont différents un test exhaustif de tous les codes possibles est réalisable. En effet, les codeurs convolutifs utilisés dans les turbo-codes ont, habituellement, une mémoire d'au plus 3 or il existe seulement 36 codeurs de type $(2, 1)$ avec une mémoire inférieure ou égale à 3 respectant l'hypothèse numéro 2. Cette seconde hypothèse est satisfaite par tous les codeurs utilisés en pratique, un codeur ne vérifiant pas cette hypothèse émet au moins un bit ne dépendant pas de l'information, ce bit est donc inutile au décodage. Enfin, la troisième hypothèse peut être légèrement moins stricte. En effet, il suffit de connaître l'initialisation des registres au début du codage de chaque mot, celle-ci peut donc être égale à 0 pour chacun des mots ou varier d'un mot à un autre. Cependant il est préférable de connaître cette initialisation.

5.2.2 Principe de la méthode et calculs des probabilités

Le but est de retrouver la permutation qui est la plus vraisemblablement celle utilisée connaissant l'ensemble de mots observés en sortie de canal. Autrement dit, on souhaite trouver la permutation $\tilde{\pi}$ maximisant la probabilité :

$$\boldsymbol{P}(\pi = \tilde{\pi} | \tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M)$$

Pour cela, la permutation est reconstruite pas à pas en maximisant, à chaque instant $t \in \{1, \ldots, N\}$, la probabilité :

$$\boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}^1_{1..N}, \tilde{\boldsymbol{w}}^1_{1..t}, \tilde{\boldsymbol{u}}^2_{1..N}, \tilde{\boldsymbol{w}}^2_{1..t}, \ldots, \tilde{\boldsymbol{u}}^M_{1..N}, \tilde{\boldsymbol{w}}^M_{1..t})$$

La proposition 5.1 permet de simplifier le calcul de cette probabilité en la décomposant en un produit de probabilités facilement calculables :

Proposition 5.1.

$$\begin{aligned}\boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}^1_{1..N}, \tilde{\boldsymbol{w}}^1_{1..t}, \tilde{\boldsymbol{u}}^2_{1..N}, \tilde{\boldsymbol{w}}^2_{1..t}, \ldots, \tilde{\boldsymbol{u}}^M_{1..N}, \tilde{\boldsymbol{w}}^M_{1..t}) \\ = \rho \prod_{s=1}^{M} \boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}^s_{1..N}, \tilde{\boldsymbol{w}}^s_{1..t})\end{aligned}$$

Où $\rho \overset{def}{=} \frac{1}{\sum_{j=1}^{N} \rho_j}$ *et* $\rho_j \overset{def}{=} \prod_{s=1}^{M} \boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}^s_{1..N}, \tilde{\boldsymbol{w}}^s_{1..t})$

Démonstration. Afin de simplifier les écritures, les notations suivantes sont utilisées :

- E_t l'événement $\tilde{\pi}(1) = p_1, \ldots, \tilde{\pi}(t-1) = p_{t-1}$ où $(p_1, \ldots, p_{t-1})$ est un $(t-1)$-uplet de $\{1, \ldots, N\}^{t-1}$.
- $F_{t,s}$ l'événement $\tilde{\boldsymbol{u}}^s_{1..N}$ et $\tilde{\boldsymbol{w}}^s_{1..t}$ sont égaux aux valeurs observées.

Avec ces notations, la probabilité à décomposer s'écrit : $\boldsymbol{P}(\pi(t) = j | E_t, F_{t,1}, \ldots, F_{t,M})$

L'hypothèse d'indépendance des mots de code implique :

$$\boldsymbol{P}(F_{t,1}, \ldots, F_{t,M} | E_t, \pi(t) = j) = \prod_{s=1}^{M} \boldsymbol{P}(F_{t,s} | E_t, \pi(t) = j)$$

D'où :

$$\begin{aligned}\boldsymbol{P}(\pi(t)=j|E_t,F_{t,1},\ldots,F_{t,M}) &= \frac{\boldsymbol{P}(F_{t,1},\ldots,F_{t,M}|E_t,\pi(t)=j)\boldsymbol{P}(E_t,\pi(t)=j)}{\boldsymbol{P}(E_t,F_{t,1},\ldots,F_{t,M})}\\ &= \frac{\prod_{s=1}^{M}\boldsymbol{P}(F_{t,s}|E_t,\pi(t)=j)\boldsymbol{P}(E_t,\pi(t)=j)}{\boldsymbol{P}(E_t,F_{t,1},\ldots,F_{t,M})}\\ &= \frac{\prod_{s=1}^{M}\boldsymbol{P}(\pi(t)=j|E_t,F_{t,s})\prod_{s=1}^{M}\boldsymbol{P}(E_t,F_{t,s})\boldsymbol{P}(E_t,\pi(t)=j)}{\prod_{s=1}^{M}\boldsymbol{P}(E_t,\pi(t)=j)\boldsymbol{P}(E_t,F_{t,1},\ldots,F_{t,M})}\end{aligned}$$

L'hypothèse sur le choix uniforme de la permutation, entraîne que $\boldsymbol{P}(E_t,\pi(t)=j)$ ne dépend pas de j. Par conséquent, la quantité ci-dessous ne dépend pas de la valeur de j.

$$\frac{\prod_{s=1}^{M}\boldsymbol{P}(E_t,F_{t,s})\boldsymbol{P}(E_t,\pi(t)=j)}{\prod_{s=1}^{M}\boldsymbol{P}(E_t,\pi(t)=j)\boldsymbol{P}(E_t,F_{t,1},\ldots,F_{t,M})}$$

On en déduit :

$$\begin{aligned}&\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}^1_{1..N},\tilde{\boldsymbol{w}}^1_{1..t},\ldots,\tilde{\boldsymbol{u}}^M_{1..N},\tilde{\boldsymbol{w}}^M_{1..t})\\ &\quad=\rho\prod_{s=1}^{M}\boldsymbol{P}(\pi(t)=j|E_t,F_{t,s})\\ &\quad=\rho\prod_{s=1}^{M}\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}^s_{1..N},\tilde{\boldsymbol{w}}^s_{1..t}),\end{aligned}$$

où ρ est une constante de normalisation telle que :

$$\sum_{j=1}^{N}\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}^1_{1..N},\tilde{\boldsymbol{w}}^1_{1..t},\ldots,\tilde{\boldsymbol{u}}^M_{1..N},\tilde{\boldsymbol{w}}^M_{1..t})=1.$$

Il s'en déduit l'égalité :

$$\rho=\frac{1}{\sum_{j=1}^{N}\rho_j}\text{ avec }\rho_j=\prod_{s=1}^{M}\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}^s_{1..N},\tilde{\boldsymbol{w}}^s_{1..t})$$

□

Quant à la proposition 5.2 elle décompose le calcul de chacun des termes du produit précédent en fonction des probabilités déduites des observations en sortie de canal. Les calculs de cette proposition font appel à des valeurs utilisées lors de l'algorithme de décodage BCJR (présenté dans la sous-section 2.3.2) pour déterminer les probabilités de chacun des états internes du codeur.

Notations. $\sum_{(a,b,\beta):\alpha\overset{ab}{\to}\beta}$ représente la somme sur tous les triplets possibles (a,b,β), tels que si le codeur est à l'état α il passe à l'état β en entrant a et il émet alors b pour bit de redondance.

Proposition 5.2. *Soit e_t l'état interne du codeur $\mathcal{C}_2$ après que les t premiers bits d'information permutés sont entrés dans ce codeur. Soit la fonction f définie par :*

$$f(\alpha,a,b,t,j) \stackrel{def}{=} \boldsymbol{P}(\tilde{u}_j|u_j=a)\boldsymbol{P}(\tilde{w}_t|w_t=b)\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1})$$

Alors :

$$\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t}) = \gamma\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\sum_{\alpha\in\{0,1\}^m}\sum_{(a,b,\beta):\alpha\stackrel{ab}{\to}\beta} f(\alpha,a,b,t,j) \quad (5.1)$$

Ici γ est une constante de normalisation donnée par :

$$\gamma \stackrel{def}{=} \frac{\boldsymbol{P}(\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\pi(t)=j)\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1))}{\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t},\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1))}\prod_{l\notin\{\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1)\}}\boldsymbol{P}(\tilde{u}_l)$$

Démonstration. La notation E_t utilisée dans la proposition 5.1 est reprise, elle représente l'événement $\tilde{\pi}(1)=p_1,\ldots,\tilde{\pi}(t-1)=p_{t-1}$, où $(p_1,\ldots,p_{t-1})$ est un $(t-1)$-uplet d'éléments distincts de $\{1,\ldots,N\}$. Il se déduit :

$$\begin{aligned}
&\boldsymbol{P}(\pi(t)=j|E_t,\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t})\\
&\qquad=\frac{\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\boldsymbol{P}(E_t,\pi(t)=j)}{\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t},E_t)}\\
&\qquad=\eta_1\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j),
\end{aligned}$$

où

— Si $j\in\{\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1)\}$ alors $\eta_1=0$.

— Sinon η_1 est une quantité qui ne dépend pas de j. Ceci vient du fait que $\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t},E_t)$ ne dépend pas de j et du fait que π est supposée choisie uniformément et aléatoirement ce qui implique que $\boldsymbol{P}(E_t,\pi(t)=j)$ ne dépend pas de la valeur de j.

On en déduit les égalités suivantes :

$$\begin{aligned}
&\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\\
&\qquad=\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{u}}_{\tilde{\pi}(t+1)..\tilde{\pi}(N)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\\
&\qquad=\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(t+1)..\tilde{\pi}(N)}|E_t,\pi(t)=j,\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t})\\
&\qquad=\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\Pi_{l\notin\{\pi(1),\ldots,\pi(t-1),j\}}\boldsymbol{P}(\tilde{u}_l) \quad (5.2)\\
&\qquad=\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\Pi_{l\notin\{\pi(1),\ldots,\pi(t-1)\}}\boldsymbol{P}(\tilde{u}_l)\\
&\qquad=\eta_2\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j),
\end{aligned}$$

où η_2 est une constante qui ne dépend pas de j. La quantité $\Pi_{l\notin\{\pi(1),\ldots,\pi(t-1),j\}}\boldsymbol{P}(\tilde{u}_l)$ de l'équation (5.2) utilise le fait que les $\tilde{u}_l$ sont indépendants de $\tilde{u}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}$, $\tilde{u}_j$ et $\tilde{\boldsymbol{w}}_{1..t}$ lorsque $l\notin\{\pi(1),\ldots,\pi(t-1),j\}$, par hypothèse du choix uniforme et aléatoire de l'information.

Il faut donc déterminer $\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)$:

$$\begin{aligned}
&\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)},\tilde{\boldsymbol{w}}_{1..t}|E_t,\pi(t)=j)\\
&\qquad=\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}|E_t,\pi(t)=j)\boldsymbol{P}(\tilde{u}_{\tilde{\pi}(t)},\tilde{w}_t|E_t,\pi(t)=j,\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1})\\
&\qquad=\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}|E_t)\boldsymbol{P}(\tilde{u}_j,\tilde{w}_t|E_t,\pi(t)=j,\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1})\\
&\qquad=\eta_3\boldsymbol{P}(\tilde{u}_j,\tilde{w}_t|E_t,\pi(t)=j,\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1})
\end{aligned}$$

Il peut être remarqué que la quantité $\eta_3 = \boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}|E_t)$ ne dépend pas de j. Il faut maintenant déterminer la probabilité :

$$A(j) \stackrel{\text{def}}{=} \boldsymbol{P}(\tilde{u}_j, \tilde{w}_t|E_t, \pi(t) = j, \tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..}\tilde{\boldsymbol{u}}_{(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1})$$

En notant e_t l'état interne du codeur $\mathcal{C}_2$ à l'instant t, c'est-à-dire après avoir entré $\boldsymbol{u}_{\pi(1)..\pi(t)}$ dans celui-ci, il se déduit les égalités suivantes. Pour rappel, $e_t \in \{0,1\}^m$ où m est le nombre de registres du codeur $\mathcal{C}_2$.

$$\begin{aligned} A(j) &= \sum_{\alpha\in\{0,1\}^m} \boldsymbol{P}(e_{t-1} = \alpha|E_t, \pi(t) = j, \tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}) \\ &\qquad \boldsymbol{P}(\tilde{u}_j, \tilde{w}_t|e_{t-1} = \alpha, E_t, \pi(t) = j, \tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}) \\ &= \sum_{\alpha\in\{0,1\}^m} \boldsymbol{P}(\tilde{u}_j, \tilde{w}_t|e_{t-1} = \alpha)\boldsymbol{P}(e_{t-1} = \alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}) \qquad (5.3) \\ &= \sum_{\alpha\in\{0,1\}^m} \sum_{a,b,\beta:\alpha\stackrel{ab}{\to}\beta} f(\alpha, a, b, t, j) \end{aligned}$$

Ici $f(\alpha, a, b, t, j) \stackrel{\text{def}}{=} \boldsymbol{P}(\tilde{u}_j|u_j = a)\boldsymbol{P}(\tilde{w}_t|w_t = b)\boldsymbol{P}(e_{t-1} = \alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1})$.
L'égalité (5.3) découle du fait que $\tilde{u}_j$ et $\tilde{w}_t$ sont liés à $\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}$ et $\tilde{\boldsymbol{w}}_{1..t-1}$ uniquement par e_{t-1}.

En reprenant les calculs précédents on en déduit :

$$\begin{aligned} &\boldsymbol{P}(\pi(t) = j|E_t, \tilde{\boldsymbol{u}}_{1..N}, \tilde{\boldsymbol{w}}_{1..t}) \\ &\quad = \eta_1\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N}, \tilde{\boldsymbol{w}}_{1..t}|E_t, \pi(t) = j) \\ &\quad = \eta_1\eta_2\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)}, \tilde{\boldsymbol{w}}_{1..t}|E_t, \pi(t) = j) \\ &\quad = \eta_1\eta_2\eta_3\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\boldsymbol{P}(\tilde{u}_j, \tilde{w}_t|E_t, \pi(t) = j, \tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}) \\ &\quad = \eta_1\eta_2\eta_3\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\sum_{\alpha\in\{0,1\}^m}\sum_{a,b,\beta:\alpha\stackrel{ab}{\to}\beta} f(\alpha, a, b, t, j) \\ &\quad = \gamma\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\sum_{\alpha\in\{0,1\}^m}\sum_{a,b,\beta:\alpha\stackrel{ab}{\to}\beta} f(\alpha, a, b, t, j), \end{aligned}$$

où

$$\gamma \stackrel{\text{def}}{=} \frac{\boldsymbol{P}(E_t, \pi(t) = j)}{\boldsymbol{P}(\tilde{\boldsymbol{u}}_{1..N}, \tilde{\boldsymbol{w}}_{1..t}, E_t)}\boldsymbol{P}(\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)}, \tilde{\boldsymbol{w}}_{1..t-1}|E_t)\prod_{l\notin\{\tilde{\pi}(1),...,\tilde{\pi}(t-1)\}}\boldsymbol{P}(\tilde{u}_l)$$

est une quantité qui ne dépend pas de j. □

Dans l'expression de $f(\alpha, a, b, t, j)$ les probabilités $\boldsymbol{P}(\tilde{u}_j|u_j = a)$ et $\boldsymbol{P}(\tilde{w}_t|w_t = b)$ dépendent du type de canal utilisé, les équations suivantes permettent d'exprimer $f(\alpha, a, b, t, j)$ en fonction des probabilités déduites des observations en sortie de canal. Ceci permet d'obtenir

un algorithme de reconstruction ne dépendant pas du type de canal utilisé, mais uniquement des probabilités déduites des observations.

$$\begin{aligned} f(\alpha,a,b,t,j) &= \boldsymbol{P}(\tilde{u}_j|u_j=a)\boldsymbol{P}(\tilde{w}_t|w_t=b)\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}) \\ &= \frac{\boldsymbol{P}(u_j=a|\tilde{u}_j)\boldsymbol{P}(\tilde{u}_j)}{\boldsymbol{P}(u_j=a)}\frac{\boldsymbol{P}(w_t=b|\tilde{w}_t)\boldsymbol{P}(\tilde{w}_t)}{\boldsymbol{P}(w_t=b)}\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}) \\ &= \eta\boldsymbol{P}(\tilde{u}_j)\boldsymbol{P}(u_j=a|\tilde{u}_j)\boldsymbol{P}(w_t=b|\tilde{w}_t)\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}), \end{aligned}$$

où $\eta = \frac{\boldsymbol{P}(\tilde{w}_t)}{\boldsymbol{P}(u_j=a)\boldsymbol{P}(w_t=b)}$ est une quantité qui ne dépend ni de j ni de a ni de b. En effet, $\boldsymbol{P}(u_j=a)$ et $\boldsymbol{P}(w_t=b)$ sont constantes quelles que soit les valeurs de j, a et b par hypothèse du choix aléatoire et suivant une distribution uniforme des mots d'information. En remplaçant l'expression de $f(\alpha,a,b,t,j)$ dans l'équation (5.1) une expression simple à calculer de la probabilité recherchée est obtenue :

$$\begin{aligned} &\boldsymbol{P}(\pi(t)=j|\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1),\tilde{\boldsymbol{u}}_{1..N},\tilde{\boldsymbol{w}}_{1..t}) \\ &= \gamma\frac{1}{\boldsymbol{P}(\tilde{u}_j)}\sum_{\alpha\in\{0,1\}^m}\sum_{a,b,\beta:\alpha\overset{ab}{\to}\beta} f(\alpha,a,b,t,j) \\ &= \gamma\eta\sum_{\alpha\in\{0,1\}^m}\sum_{a,b,\beta:\alpha\overset{ab}{\to}\beta}\boldsymbol{P}(u_j=a|\tilde{u}_j)\boldsymbol{P}(w_t=b|\tilde{w}_t)\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t-1)},\tilde{\boldsymbol{w}}_{1..t-1}) \\ &= \gamma_2\sum_{\alpha\in\{0,1\}^m}\sum_{a,b,\beta:\alpha\overset{ab}{\to}\beta}\boldsymbol{P}(u_j=a|\tilde{u}_j)\boldsymbol{P}(w_t=b|\tilde{w}_t)\boldsymbol{P}(e_{t-1}=\alpha|\tilde{\boldsymbol{u}}_{\pi(1)..\tilde{\pi}(t-1)}\tilde{\boldsymbol{w}}_{1..t-1}) \end{aligned}$$

5.2.3 Algorithme de reconstruction de la permutation

Des différents calculs de la sous-section précédente, il découle l'algorithme 3 qui détermine à chaque instant t la valeur la plus probable de $\pi(t)$ connaissant $\tilde{\boldsymbol{u}}^1_{1..N}$, $\tilde{\boldsymbol{w}}^1_{1..t}$, $\ldots$, $\tilde{\boldsymbol{u}}^M_{1..N}$, $\tilde{\boldsymbol{w}}^M_{1..t}$ et $\tilde{\pi}(1),\ldots,\tilde{\pi}(t-1)$. La complexité de cet algorithme est, par construction, en $\mathcal{O}(2^mN^2M)$, où m est la mémoire du code convolutif $\mathcal{C}_2$.

L'annexe B.1 détaille la reconstruction de la permutation sur un exemple très simple et sans calcul de probabilité.

5.2.4 Étude de la probabilité de reconstruire la permutation

L'étude de la probabilité de reconstruire correctement la permutation utilisée dans le turbo-code est effectuée sur un exemple très simple afin de simplifier les différents calculs. Les hypothèses sont les suivantes :

- Le canal de transmission est un canal à effacement de probabilité p. L'observation d'un effacement est noté ?.
- Le codeur $\mathcal{C}_2$, schématisé sur la figure 5.2, satisfait les équations $w_t = x_t \oplus e_{t-1}$ et $e_t = e_{t-1} \oplus x_t$ d'où $e_t = w_t$.

Algorithme 3 : RECONSTRUCTION_PERMUTATION_TURBO-CODE_PARALLÈLE.

Entrées :

1. Les probabilités $\boldsymbol{P}(u_t^s = a|\tilde{u}_t^s)$ et $\boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)$ pour tout $t \in \{1, \ldots, N\}$, tous $s \in \{1, \ldots, M\}$ et toutes les valeurs binaires a et b.
2. Le deuxième codeur convolutif $\mathcal{C}_2$ de rendement $\frac{1}{2}$, et m sa mémoire.

Sortie : Une permutation $\tilde{\pi}$, de taille N, qui maximise, pour tout $t \in \{1, \ldots, N\}$, $\boldsymbol{P}(\pi(t) = j|\tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^1, \tilde{\boldsymbol{w}}_{1..t}^1, \ldots, \tilde{\boldsymbol{u}}_{1..N}^M, \tilde{\boldsymbol{w}}_{1..t}^M)$

Initialisation des probabilités des états internes :
Pour $s = 1, \ldots, M$
  $p[s][0] \leftarrow 1$ //Les registres sont supposés initialisés à 0
  Pour $\alpha \in \{1, \ldots, 2^m - 1\}$
    $p[s][\alpha] \leftarrow 0$

Pour $t = 1, \ldots, N$
  Recherche de la valeur la plus probable de $\pi(t)$:
  Pour $s = 1, \ldots, M$
    Pour $j \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(t-1)\}$
      $q[s][j] \leftarrow \sum_{\alpha \in \{0,1\}^m} \sum_{(a,b,\beta):\alpha \overset{ab}{\to} \beta} \boldsymbol{P}(u_j^s = a|\tilde{u}_j^s)\boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)p[s][\alpha]$
      $//q[s][j] = \mu_1 \boldsymbol{P}(\pi(t) = j|\tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^s, \tilde{\boldsymbol{w}}_{1..t}^s)$ avec μ_1 une constante de normalisation dépendant de s
  Pour $j \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(t-1)\}$
    $r[j] \leftarrow \prod_{s=1}^M q[s][j]$
    $//r[j] = \mu_2 \boldsymbol{P}(\pi(t) = j|\tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^1, \tilde{\boldsymbol{w}}_{1..t}^1, \ldots, \tilde{\boldsymbol{u}}_{1..N}^M, \tilde{\boldsymbol{w}}_{1..t}^M)$ avec μ_2 une constante de normalisation.
  $\tilde{\pi}(t) \leftarrow \arg\max_j r[j]$

  Mise à jour des probabilités des états internes :
  Pour $s = 1, \ldots, M$
    Pour $\beta \in \{0,1\}^m$
      $q[s][\beta] \leftarrow \sum_{\alpha:\alpha \overset{ab}{\to} \beta} p[s][\alpha]\boldsymbol{P}(u_{\tilde{\pi}(t)}^s = a|\tilde{u}_{\tilde{\pi}(t)}^s)\boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)$
    Pour $\beta \in \{0,1\}^m$
      $p[s][\beta] \leftarrow \frac{q[s][\beta]}{\sum_\alpha q[s][\alpha]}$
      $//p[s][\beta] = \boldsymbol{P}(e_t^s = \beta|\tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(t)}^s, \tilde{\boldsymbol{w}}_{1..t}^s)$
Retourner $\tilde{\pi}$

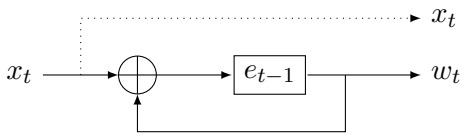


FIGURE 5.2 – Codeur $\mathcal{C}_2$ considéré pour l'étude de la probabilité de reconstruire correctement la permutation

En reprenant les notations des turbo-codes parallèles (figure 5.1), ces équations mènent aux égalités : $w_t = u_{\pi(t)} \oplus e_{t-1}$ et $e_t = e_{t-1} \oplus u_{\pi(t)}$. La sortie $\boldsymbol{x}$ n'étant pas transmise sur le canal se sont ces dernières équations qui vont être utilisées afin de reconstruire la permutation.

La permutation π étant supposée choisie de façon uniforme parmi toutes les permutations de taille N, la probabilité P de reconstruire correctement la permutation π se décompose en un produit de probabilités :

$$\begin{aligned} P &= \prod_{t=1}^{N} \boldsymbol{P}(\pi(t) = \tilde{\pi}(t) | \pi(1) = \tilde{\pi}(1), \ldots, \pi(t-1) = \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^{1}, \tilde{\boldsymbol{w}}_{1..t}^{1}, \ldots, \tilde{\boldsymbol{u}}_{1..N}^{M}, \tilde{\boldsymbol{w}}_{1..t}^{M}) \\ &= \prod_{t=1}^{N} \Big(1 - \boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | \pi(1) = \tilde{\pi}(1), \ldots, \pi(t-1) = \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^{1}, \tilde{\boldsymbol{w}}_{1..t}^{1}, \ldots, \tilde{\boldsymbol{u}}_{1..N}^{M}, \tilde{\boldsymbol{w}}_{1..t}^{M})\Big) \end{aligned}$$

Notations. Pour alléger les écritures, les notations suivantes sont utilisées :
— $\pi_{1..t-1}$ représente l'événement "$\pi(1) = \tilde{\pi}(1), \ldots, \pi(t-1) = \tilde{\pi}(t-1)$"
— $\tilde{\boldsymbol{u}}_{1..N}^{1..M}$ représente l'événement "$\tilde{\boldsymbol{u}}_{1..N}^{1}, \ldots, \tilde{\boldsymbol{u}}_{1..N}^{M}$ ont été observés en sortie de canal"
— $\tilde{\boldsymbol{w}}_{1..t}^{1..M}$ représente l'événement "$\tilde{\boldsymbol{w}}_{1..t}^{1}, \ldots, \tilde{\boldsymbol{w}}_{1..t}^{M}$ ont été observés en sortie de canal"

Afin de déterminer les probabilités $\boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M})$ pour tout t, quelques calculs intermédiaires sont nécessaires.

Sur le canal à effacement, l'algorithme de reconstruction de la permutation peut être vu comme l'élimination des indices impossibles c'est-à-dire de l'ensemble des indices j tels que $\boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^{1}, \tilde{\boldsymbol{w}}_{1..t}^{1}, \ldots, \tilde{\boldsymbol{u}}_{1..N}^{M}, \tilde{\boldsymbol{w}}_{1..t}^{M}) = 0$ et la conservation des autres. Pour le codeur convolutif considéré, cette probabilité est nulle s'il existe un mot $\tilde{\boldsymbol{m}}^i$ tel que $\tilde{u}_j^i$, e_{t-1}^i et $\tilde{w}_t^i$ ne sont pas des effacements et sont tels que $\tilde{u}_j^i \neq e_{t-1}^i \oplus \tilde{w}_t^i$. Ensuite, parmi les indices conservés l'un d'entre eux est choisi. Ce choix s'effectue suivant la proposition suivante :

Proposition 5.3. *Sur le canal à effacement l'algorithme 3 choisit, à l'instant t, l'indice j tel que $\boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^{1}, \tilde{\boldsymbol{w}}_{1..t}^{1}, \ldots, \tilde{\boldsymbol{u}}_{1..N}^{M}, \tilde{\boldsymbol{w}}_{1..t}^{M}) \neq 0$ et maximisant :*

$$\#\{i \in \{1, \ldots, M\}, \tilde{u}_j^i \text{ et } \tilde{w}_t^i \text{ ne sont pas des effacements et } e_t^i \text{ est connu}\}$$

où $\#$ désigne le cardinal de l'ensemble.

Démonstration. A l'instant t, l'algorithme choisit l'indice qui maximise la probabilité $r[j] = \prod_{s=1}^{M} q[s][j]$ avec

$$q[s][j] = \sum_{\alpha \in \{0,1\}^m} \sum_{(a,b,\beta): \alpha \xrightarrow{ab} \beta} \boldsymbol{P}(u_j^s = a | \tilde{u}_j^s) \boldsymbol{P}(w_t^s = b | \tilde{w}_t^s) p[s][\alpha]$$

et $p[s][\alpha]$ est la probabilité que le codeur soit à l'état α connaissant $\pi_{1..t-1}$, $\tilde{\boldsymbol{u}}^s_{1..N}$ et $\tilde{\boldsymbol{w}}^s_{1..t}$.

Par hypothèse, le codeur convolutif $\mathcal{C}_2$ est un codeur binaire de type $(2,1)$, seules les valeurs 0 et 1 peuvent être entrées dans le codeur. Il y a donc exactement deux changements d'état possibles en partant d'un état donné. Autrement dit, la probabilité $q[s][j]$ est égale à :

$$q[s][j] = \sum_{\alpha\in\{0,1\}^m} \Big(\boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j)\boldsymbol{P}(w^s_t = b_{\alpha,0}|\tilde{w}^s_t)p[s][\alpha] + \boldsymbol{P}(u^s_j = 1|\tilde{u}_j)\boldsymbol{P}(w^s_t = b_{\alpha,1}|\tilde{w}^s_t)p[s][\alpha]\Big),$$

où $b_{\alpha,i}$ représente la redondance émise par le codeur en entrant i lorsqu'il était à l'état α.

Dans les hypothèses de départ, il est supposé que le premier bit de redondance émis par $\mathcal{C}_2$ dépend du premier bit entrant dans le codeur. Cette hypothèse implique que pour un état α donné, $b_{\alpha,0} \neq b_{\alpha,1}$. Et plus précisément, pour le codeur considéré $b_{0,0} = 0$ et donc $b_{0,1} = 1$.

L'initialisation des registres du codeur est supposée connue et égale à 0 d'où $p[s][0] = 1$ et $p[s][\alpha] = 0, \forall\alpha \neq 0$. Ce qui entraîne qu'à l'instant $t = 1$:

$$\begin{aligned} q[s][j] &= \boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j)\boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1) + \boldsymbol{P}(u^s_j = 1|\tilde{u}^s_j)\boldsymbol{P}(w^s_1 = 1|\tilde{w}^s_1) \\ &= \boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j)\boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1) + (1 - \boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j))(1 - \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \end{aligned}$$

Le canal considéré est le canal à effacement, d'où $\boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j)$ et $\boldsymbol{P}(w^s_j = 0|\tilde{w}^s_j)$ sont à valeurs dans $\{0, 1, \frac{1}{2}\}$.

Afin de comprendre le choix effectué par l'algorithme, supposons que deux indices j et k sont conservés.
L'indice j est conservé signifie que $r[j] \neq 0$, autrement dit :

$$\boldsymbol{P}(\pi(1) = j|\tilde{\boldsymbol{u}}^1_{1..N}, \tilde{\boldsymbol{w}}^1_1, \ldots, \tilde{\boldsymbol{u}}^M_{1..N}, \tilde{\boldsymbol{w}}^M_1) \neq 0$$

ce qui implique que $\forall s \in \{1, \ldots, M\}$, $q[s][j] \neq 0$ et donc que

$$(\boldsymbol{P}(u^s_j = 0|\tilde{u}^s_j), \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \notin \{(0,1),(1,0)\}$$

Il en est de même pour conserver l'indice k.

Les quatre quantités suivantes vont être utilisées : pour $i \in \{j, k\}$

$$\begin{aligned} a_i &= \#\{s \in \{1,\ldots,M\}, (\boldsymbol{P}(u^s_i = 0|\tilde{u}^s_i), \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \in \{(0,0),(1,1)\}\} \\ b_i &= \#\{s \in \{1,\ldots,M\}, (\boldsymbol{P}(u^s_i = 0|\tilde{u}^s_i), \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \in \{(\frac{1}{2},0),(\frac{1}{2},1)\}\} \\ c_i &= \#\{s \in \{1,\ldots,M\}, (\boldsymbol{P}(u^s_i = 0|\tilde{u}^s_i), \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \in \{(0,\frac{1}{2}),(1,\frac{1}{2})\}\} \\ d_i &= \#\{s \in \{1,\ldots,M\}, (\boldsymbol{P}(u^s_i = 0|\tilde{u}^s_i), \boldsymbol{P}(w^s_1 = 0|\tilde{w}^s_1)) \in \{(\frac{1}{2},\frac{1}{2})\}\} \end{aligned}$$

Par construction, les égalités $a_j + b_j = a_k + b_k$ et $c_j + d_j = c_k + d_k$ sont vérifiées.

Avec ces notations, $r[j]$ s'écrit :

$$\begin{aligned} r[j] &= \prod_{s=1}^{M} q[s][j] \\ &= 1^{a_j} \times \left(\frac{1}{2}\right)^{b_j} \times \left(\frac{1}{2}\right)^{c_j} \times \left(\frac{1}{2}\right)^{d_j} \\ &= 2^{-b_j-c_j-d_j} \end{aligned}$$

L'algorithme choisit la valeur i maximisant $r[i]$, pour savoir lequel des indices j ou k est choisi il faut donc résoudre l'inégalité $r[j] > r[k]$.

$$\begin{aligned} r[j] > r[k] &\Leftrightarrow \frac{r[j]}{r[k]} > 1 \\ &\Leftrightarrow \frac{2^{-b_j-c_j-d_j}}{2^{-b_k-c_k-d_k}} > 1 \\ &\Leftrightarrow 2^{b_k-b_j} > 1 \text{ car } c_j + d_j = c_k + d_k \\ &\Leftrightarrow b_j < b_k \\ &\Leftrightarrow a_j > a_k \text{ car } a_j + b_j = a_k + b_k \end{aligned}$$

L'indice j est donc choisi s'il maximise la quantité a, c'est-à-dire si c'est l'indice pour lequel $\boldsymbol{P}(u_j^s = 0|\tilde{u}_j^s) = \boldsymbol{P}(w_1^s = 0|\tilde{w}_1^s)$ le plus souvent.

Remarque : Si le codeur considéré était tel que $b_{0,0} = 1$ et donc que $b_{0,1} = 0$, alors il aurait été impossible d'avoir $\boldsymbol{P}(u_j^s = 0|\tilde{u}_j^s) = \boldsymbol{P}(w_1^s = 0|\tilde{w}_1^s)$ (sinon l'indice aurait été éliminé). Le même type de calcul conduit à choisir l'indice qui vérifie le plus de fois :

$$(\boldsymbol{P}(u_j^s = 0|\tilde{u}_j^s), \boldsymbol{P}(w_1^s = 0|\tilde{w}_1^s)) \in \{(0,1),(1,0)\}$$

En fait, quelle que soit la valeur de $b_{0,0}$, l'algorithme choisit l'indice conservé qui maximise

$$\#\{s \in \{1,\ldots,M\}, \tilde{u}_j^s \text{ et } \tilde{w}_1^s \text{ ne sont pas des effacements}\}$$

Ce choix est en fait naturel, puisqu'il correspond au choix de l'indice non éliminé qui à le plus faible risque d'avoir été conservé par erreur.

Ce raisonnement considérant seulement 2 indices conservés est facilement généralisable à un nombre d'indices conservés plus important.

Les mêmes calculs peuvent être effectués quel que soit l'instant t considéré. Cependant lorsque $t \neq 1$, il peut arriver que pour certains mots l'état des registres du codeur ne soit pas connu, ces mots sont dits "inutiles", c'est-à-dire qu'ils n'apporteront aucune information et donc ne serviront pas à départager deux indices potentiels. En effet, par exemple pour un codeur avec un seul registre, lorsque l'état interne est inconnu : $p[s][0] = p[s][1] = 1/2$ d'où :

$$\begin{aligned} q[s][j] &= \sum_{\alpha\in\{0,1\}} \frac{1}{2}\Big(\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{\alpha,0}|\tilde{w}_t^s)+\boldsymbol{P}(u_j^s=1|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{\alpha,1}|\tilde{w}_t^s)\Big) \\ &= \sum_{\alpha\in\{0,1\}} \frac{1}{2}\Big(\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{\alpha,0}|\tilde{w}_t^s)+(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s))(1-\boldsymbol{P}(w_t^s=b_{\alpha,0}|\tilde{w}_t^s))\Big) \\ &= \sum_{\alpha\in\{0,1\}} \frac{1}{2}\Big(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)-\boldsymbol{P}(w_t^s=b_{\alpha,0}|\tilde{w}_t^s)+2\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{\alpha,0}|\tilde{w}_t^s)\Big) \end{aligned}$$

Le codeur ayant un seul registre, $b_{0,0} \neq b_{1,0}$, sinon la redondance ne dépendrait pas de l'état interne du codeur, il s'agirait d'un simple code à répétition. Par conséquent $q[s][j]$ ne dépend pas de j, en effet :

$$\begin{aligned} q[s][j] &= \frac{1}{2}\Big(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)-\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s)+2\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s)\Big) \\ &\quad +\frac{1}{2}\Big(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)-\boldsymbol{P}(w_t^s=b_{1,0}|\tilde{w}_t^s)+2\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{1,0}|\tilde{w}_t^s)\Big) \\ &= \frac{1}{2}\Big(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)-\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s)+2\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s)\Big) \\ &\quad +\frac{1}{2}\Big(1-\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)-(1-\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s))+2\boldsymbol{P}(u_j^s=0|\tilde{u}_j^s)(1-\boldsymbol{P}(w_t^s=b_{0,0}|\tilde{w}_t^s))\Big) \\ &= \frac{1}{2} \end{aligned}$$

Le résultat de la proposition s'en déduit : sur le canal à effacement, l'algorithme choisit, à l'instant t, l'indice non éliminé maximisant

$$\#\{s\in\{1,\dots,M\}, \tilde{u}_j^s \text{ et } \tilde{w}_t^s \text{ ne sont pas des effacements et } e_t^s \text{ est connu}\}$$

□

Les probabilités que $\tilde{u}_j^i$ et $\tilde{w}_t^i$ ne soient pas des effacements sont connues et sont toutes deux égales à $1-p$. Le lemme 5.4 détermine la probabilité s_t de ne pas connaître e_t^i. Il peut être remarqué que cette probabilité ne dépend pas du mot considéré mais uniquement de t.

Lemme 5.4. *Pour un mot donné, la probabilité s_t de ne pas connaître l'état du registre à l'instant t est donnée par :*

$$\begin{cases} s_0 = 0 \\ s_t = (1-s_{t-1})p^2 + s_{t-1}p \ \ \forall t>0 \end{cases}$$

Démonstration. Par hypothèse, le registre est initialisé à 0 au début du codage de chaque mot. L'état du registre à l'instant $t=0$ est donc connu d'où $s_0=0$.
A l'instant $t=1$, si $\tilde{w}_1$ n'est pas un effacement, autrement dit si $\tilde{w}_1 \neq ?$, alors e_1 est connu, puisque le codeur satisfait l'égalité $e_t = w_t$ et que l'hypothèse de l'utilisation d'un canal à effacement implique que si $\tilde{w}_t \neq ?$ alors $\tilde{w}_t = \tilde{w}$. La valeur e_1 peut également se déduire de $\tilde{u}_{\pi(1)}$ en utilisant l'égalité $e_1 = e_0 \oplus u_{\pi(1)}$ puisque si $\tilde{u}_{\pi(1)} \neq ?$ alors $\tilde{u}_{\pi(1)} = u_{\pi(1)}$ et qu'à cet instant la valeur de $\pi(1)$ est déjà retrouvée. Pour ne pas connaître l'état du registre à l'instant $t=1$ il faut donc que $\tilde{w}_1$ et $\tilde{u}_{\pi(1)}$ soient des effacements, ce qui se produit avec une probabilité p^2, d'où $s_1 = p^2$.
Aux instants suivants, il est impossible de connaître l'état e_t du registre lorsque :

— e_{t-1} est connu mais que $\tilde{u}_{\pi(t)} = ?$ et que $\tilde{w}_t = ?$, ce qui se produit avec une probabilité $(1 - s_{t-1})p^2$.
— e_{t-1} est inconnu et $\tilde{w}_t = ?$, que $\tilde{u}_{\pi(t)}$ soit un effacement ou non, cette configuration arrive avec une probabilité de $s_{t-1}p$.

Dans tous les autres cas, la valeur de e_t se déduit à partir des équations du codeur. On en déduit la probabilité s_t de ne pas connaître e_t pour $t \geq 1$: $s_t = (1 - s_{t-1})p^2 + s_{t-1}p$. □

A l'instant t, les mots tels que $\tilde{w}_t$ est un effacement ou tels que l'état interne du codeur n'est pas connu n'apportent aucune information pour le choix de $\tilde{\pi}(t)$. En effet, ils ne peuvent conduire à aucune contradiction sur l'équation : $\tilde{u}_j^i = e_{t-1}^i \oplus \tilde{w}_t^i$, d'où la définition :

Définition 5.5. *Un mot $\tilde{\boldsymbol{m}}^i$ est dit "utile" à l'instant t si l'état du registre e_{t-1}^i est connu et si $\tilde{w}_t$ n'est pas un effacement. Dans le cas contraire il est dit "inutile".*

La probabilité P recherchée se décompose alors de la façon suivante :

$$P = \prod_{t=1}^{N} \Bigg(1 - \sum_{m=1}^{M} \boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ \boldsymbol{P}(m \text{ mots sont utiles } | \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \Bigg)$$

Pour que $\pi(t) \neq \tilde{\pi}(t)$ il est nécessaire que plusieurs indices soient conservés et ne pas choisir le bon parmi ceux-ci. La probabilité P est donc égale à :

$$P = \prod_{t=1}^{N} \Bigg(1 - \sum_{m=1}^{M} \sum_{k=1}^{N-t-1} \boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | k+1 \text{ indices sont conservés}, m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ \boldsymbol{P}(k+1 \text{ indices sont conservés} | m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ \boldsymbol{P}(m \text{ mots sont utiles } | \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \Bigg)$$

Les deux lemmes suivants permettent de déterminer la probabilité que m mots soient utiles à l'instant t et que $k+1$ indices soient conservés connaissant ces m mots utiles.

Lemme 5.6.

$$\begin{aligned} r(m,t) &= \boldsymbol{P}(m \textit{ mots sont utiles } | \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ &= \binom{M}{m} \left((1-s_t)(1-p)\right)^m \left(1 - (1-s_t)(1-p)\right)^{M-m} \end{aligned}$$

Démonstration. Sachant que le début de la permutation est correctement reconstruit, un mot donné $\tilde{\boldsymbol{m}}^i$ est utile à l'instant t si et seulement si il peut conduire à la contradiction $\tilde{u}_j^i \neq e_{t-1}^i \oplus \tilde{w}_t^i$. Or cette contradiction est possible uniquement si e_{t-1}^i est connu et si $\tilde{w}_t^i$ n'est pas un effacement, ce qui se produit avec une probabilité $(1 - s_t)(1 - p)$. Les mots sont indépendants et le canal est sans mémoire, la probabilité qu'exactement m mots soient utiles suit donc une loi de Bernoulli de probabilité $(1 - s_t)(1 - p)$, d'où le résultat du lemme. □

Lemme 5.7.

$$\begin{aligned} q(k,m,t) &= \boldsymbol{P}(k+1 \textit{ indices sont conservés} | m \textit{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ &= \binom{N-t}{k} \left(\frac{1+p}{2}\right)^{mk} \left(1 - \left(\frac{1+p}{2}\right)^m\right)^{N-t-k} \end{aligned}$$

Démonstration. Par définition, le mot $\tilde{\boldsymbol{m}}^i$ est utile si $e^i_{t-1} \oplus \tilde{w}^i_t$ est connu, ce mot permet d'éliminer l'indice j si $\tilde{u}^i_j \neq e^i_{t-1} \oplus \tilde{w}^i_t$. Cette contradiction se produit avec un probabilité $1/2$ lorsque $\tilde{u}^i_j$ n'est pas un effacement, d'où la probabilité que le mot utile $\tilde{\boldsymbol{m}}^i$ permette l'élimination de l'indice j : $\frac{1-p}{2}$. Il s'en déduit la probabilité qu'en utilisant uniquement ce mot l'indice j soit conservé : $\frac{1+p}{2}$.

En considérant l'ensemble des m mots utiles, un indice est conservé uniquement si chacun de ces mots permet sa conservation, d'où la probabilité qu'un indice j donné soit conservé : $\left(\frac{1+p}{2}\right)^m$.

A l'instant t, c'est-à-dire lors de la recherche de $\pi(t)$, $N-t+1$ indices de $\{1,\dots,N\}$ n'ont pas encore été choisis. Dans cet ensemble d'indices restants se trouve la valeur de $\pi(t)$ par hypothèse que la permutation est correctement reconstruite sur les $t-1$ premiers indices. Cet indice ne peut pas être éliminé et fait donc parti des indices conservés. L'hypothèse sur le choix uniforme des mots d'information entraîne l'indépendance des éliminations des différents indices, la probabilité qu'exactement k mauvais indices (autrement dit k indices différents de $\pi(t)$) soient conservés suit donc la loi de Bernoulli de probabilité $\left(\frac{1+p}{2}\right)^m$. D'où le résultat du lemme. □

Sachant que le choix de la valeur de $\tilde{\pi}(t)$ s'effectue en suivant la proposition 5.3, la probabilité

$$\boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | k+1 \text{ indices sont conservés}, m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}^{1..M}_{1..N}, \tilde{\boldsymbol{w}}^{1..M}_{1..t})$$

se calcule grâce aux lemmes suivants et en utilisant la notation :

$$\Omega_{j,t} = \#\{i \in \{1,\dots,M\}, \tilde{u}^i_j = e^i_{t-1} \oplus \tilde{w}^i_t \text{ et } \tilde{u}^i_j \text{ n'est pas un effacement}\}$$

$\Omega_{j,t}$ correspond donc au nombre de mots utiles vérifiant l'égalité $\tilde{u}_j = e_{t-1} \oplus \tilde{w}_t$.

Lemme 5.8. *A l'instant t, si $j = \pi(t)$ et que m mots sont utiles alors :*

$$\boldsymbol{P}(\Omega_{j,t} = \lambda | m \textit{ mots sont utiles et } j = \pi(t)) = \binom{m}{\lambda}(1-p)^\lambda p^{m-\lambda}$$

Démonstration. Le tableau suivant indique les probabilités de chacun des couples $(\tilde{w}^i_t \oplus e^i_{t-1}, \tilde{u}^i_j)$ pour un mot $\tilde{\boldsymbol{m}}^i$ donné et sachant que $j = \pi(t)$.

$\tilde{w}^i_t \oplus e^i_{t-1}$ / $\tilde{u}^i_j$	0	1	inconnu
0	$\frac{1}{2}(1-p)^2(1-s_t)$	impossible	$\frac{1}{2}\left((1-p)(1-s_t)p + (1-p)^2 s_t + (1-p)ps_t\right)$
1	impossible	$\frac{1}{2}(1-p)^2(1-s_t)$	$\frac{1}{2}\left((1-p)(1-s_t)p + (1-p)^2 s_t + (1-p)ps_t\right)$
?	$\frac{1}{2}p(1-p)(1-s_t)$	$\frac{1}{2}p(1-p)(1-s_t)$	$p^2(1-s_t) + p^2 s_t + ps_t(1-p)$

De ce tableau, il se déduit :

$$\boldsymbol{P}(\tilde{u}^i_j = \tilde{w}^i_t \oplus e^i_{t-1} | \tilde{\boldsymbol{m}}^i \text{ est utile et } j = \pi(t)) = 1-p$$

Le résultat du lemme découle de ce résultat et de l'hypothèse sur l'indépendance des mots d'information. □

Lemme 5.9. *A l'instant t, si $j \neq \pi(t)$ et que m mots sont utiles alors :*

$$\boldsymbol{P}(\Omega_{j,t} = \lambda | m \text{ mots sont utiles et } j \neq \pi(t)) = \binom{m}{\lambda} \frac{(1-p)^\lambda (2p)^{m-\lambda}}{(1+p)^m}$$

et

$$\boldsymbol{P}(\Omega_{j,t} < \lambda | m \text{ mots sont utiles et } j \neq \pi(t)) = \sum_{\gamma=1}^{\lambda-1} \boldsymbol{P}(\Omega_{j,t} = \gamma)$$

Démonstration. Le tableau suivant indique les probabilités de chacun des couples $(\tilde{w}_t^i \oplus e_{t-1}^i, \tilde{u}_j^i)$ pour un mot $\tilde{\boldsymbol{m}}^i$ donné et sachant que $j \neq \pi(t)$ et $j \in \{1, \ldots, N\} \backslash \{\pi(1), \ldots, \pi(t-1)\}$.

$\tilde{w}_t^i \oplus e_{t-1}^i$ / $\tilde{u}_j^i$	0	1	inconnu
0	$\frac{1}{4}(1-p)^2(1-s_t)$	$\frac{1}{4}(1-p)^2(1-s_t)$	$\frac{1}{2}\left((1-p)(1-s_t)p + (1-p)^2 s_t + (1-p)ps_t\right)$
1	$\frac{1}{4}(1-p)^2(1-s_t)$	$\frac{1}{4}(1-p)^2(1-s_t)$	$\frac{1}{2}\left((1-p)(1-s_t)p + (1-p)^2 s_t + (1-p)ps_t\right)$
?	$\frac{1}{2}p(1-p)(1-s_t)$	$\frac{1}{2}p(1-p)(1-s_t)$	$p^2(1-s_t) + p^2 s_t + ps_t(1-p)$

De ce tableau il se déduit :

$$\boldsymbol{P}(\tilde{u}_j^i = \tilde{w}_t^i \oplus e_{t-1}^i | \tilde{\boldsymbol{m}}^i \text{ est utile, } j \text{ est conservé et } j \neq \pi(t)) = \frac{1-p}{1+p}$$

Et comme précédemment les résultats du lemme découlent de cette probabilité et de l'hypothèse sur l'indépendance des mots d'information. □

Les trois probabilités déterminées dans ces deux derniers lemmes ne dépendant ni de l'indice j ni de l'instant t et afin de simplifier les écritures elles sont respectivement notées : $\boldsymbol{P}(\Omega_{\pi(t)} = \lambda)$, $\boldsymbol{P}(\Omega_{\neq\pi(t)} = \lambda)$ et $\boldsymbol{P}(\Omega_{\neq\pi(t)} < \lambda)$.

Il s'en déduit le lemme suivant :

Lemme 5.10.

$$\begin{aligned} b(k,m) &= \boldsymbol{P}(\pi(t) = \tilde{\pi}(t) | k+1 \text{ indices sont conservés}, m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \\ &= \sum_{\lambda=1}^{m} \left(\boldsymbol{P}(\Omega_{\pi(t)} = \lambda) \boldsymbol{P}(\Omega_{\neq\pi(t)} < \lambda)^k \right) \\ &\qquad + \sum_{w=1}^{k} \left(\frac{1}{w+1} \binom{k}{w} \sum_{\lambda=1}^{m} \left(\boldsymbol{P}(\Omega_{\pi(t)} = \lambda) \boldsymbol{P}(\Omega_{\neq\pi(t)} = \lambda)^w \boldsymbol{P}(\Omega_{\neq\pi(t)} < \lambda)^{k-w} \right) \right) \end{aligned}$$

Démonstration. Pour que l'indice choisi soit le bon il faut que l'une des deux conditions suivantes soient vérifiées :

- $\Omega_{\pi(t),t} > \Omega_{j,t}$ pour les k indices j conservés tels que $j \neq \pi(t)$
- Si $A = \{j$ tel que j est conservé, $j \neq \pi(t)$ et $\Omega_{\pi(t),t} = \Omega_{j,t}\}$ et que pour tout indice j conservé mais $j \notin A$ alors $\Omega_{\pi(t),t} > \Omega_{j,t}$, il faut que le choix aléatoire de l'indice parmi ceux contenus dans A soit le bon.

La probabilité souhaitée s'en déduit. □

En regroupant les résultats de ces différents lemmes il en vient la proposition :

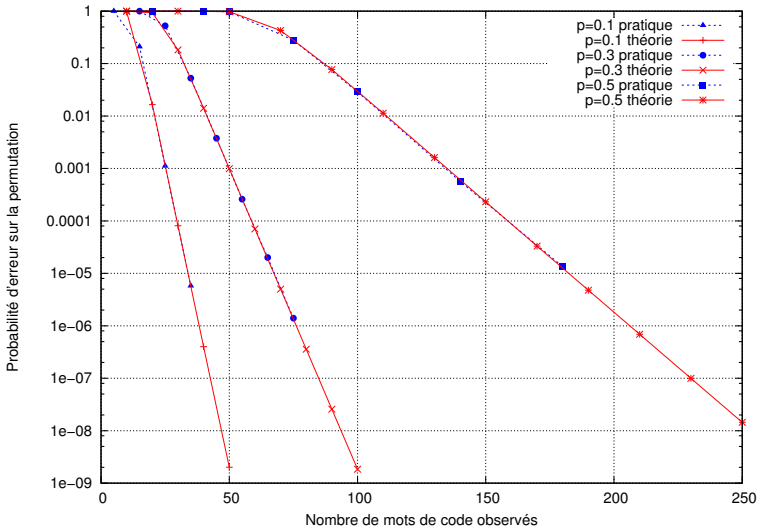


FIGURE 5.3 – Probabilité d'erreur sur la permutation reconstruite avec $N = 60$. Comparaison des probabilités observées de façon pratique (expérimentalement) avec celles obtenues par la proposition 5.11.

Proposition 5.11. *La probabilité P de reconstruire correctement la permutation π sous les hypothèses faites en début de cette sous-section est égale à :*

$$P = \prod_{t=1}^{N} \left(1 - \sum_{m=1}^{M} \sum_{k=1}^{N-t-1} (1 - b(k,m)) q(k,m,t) r(m,t) \right)$$

Démonstration. Cette proposition découle de la décomposition de la probabilité P énoncée juste avant le lemme 5.6 :

$$P = \prod_{t=1}^{N} \Bigg(1 - \sum_{m=1}^{M} \sum_{k=1}^{N-t-1} \boldsymbol{P}(\pi(t) \neq \tilde{\pi}(t) | k+1 \text{ indices sont conservés}, m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M})$$
$$\boldsymbol{P}(k+1 \text{ indices sont conservés} | m \text{ mots sont utiles}, \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M})$$
$$\boldsymbol{P}(m \text{ mots sont utiles } | \pi_{1..t-1}, \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M}) \Bigg)$$

chacune des probabilités étant remplacées par sa valeur en utilisant l'un des lemmes précédents. □

La figure 5.3 représente l'évolution de cette probabilité P (en utilisant la formule de la proposition 5.11) en fonction du nombre M de mots observés dans le cas d'une permutation de taille $N = 60$ et pour un canal à effacement de probabilité p. Il est également représenté, sur cette même figure, la probabilité obtenue de façon pratique, c'est-à-dire en effectuant de très nombreux tests afin de valider les calculs théoriques effectués dans cette sous-section.

Remarque. *Il peut être montré que pour un canal à effacement de probabilité p et un nombre M de mots suffisamment grand, la probabilité $1 - P$ de ne pas reconstruire correctement la permutation est approximée par :*

$$P_{échec} = \mathcal{O}\left(N^2 \left(\frac{2p + ((\sqrt{2}-1)p+1)^2(1-p)^2}{2(1+p^2-p)} \right)^M \right)$$

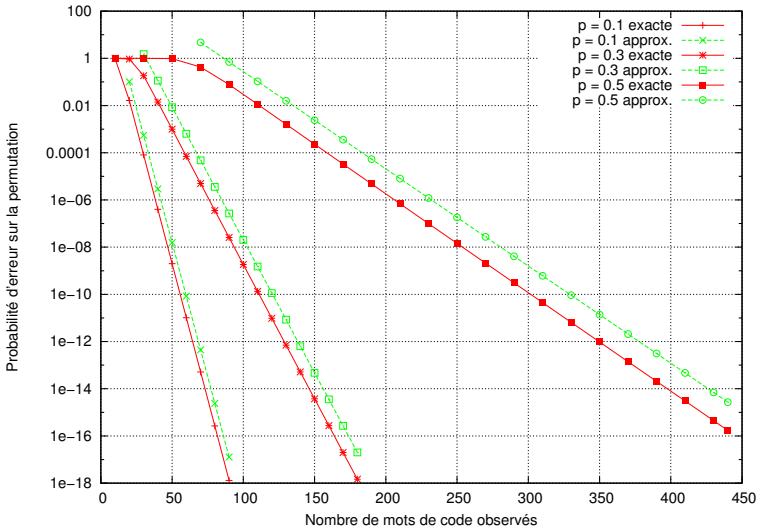


FIGURE 5.4 – Probabilité de ne pas reconstruire correctement la permutation de longueur $N = 60$. Valeurs exactes obtenues en utilisant la proposition 5.11 et son approximation donnée dans la remarque

Sur la figure 5.4, sont tracées les courbes représentant les probabilités de ne pas reconstruire correctement la permutation obtenues par cette approximation ainsi que la valeur exacte, donnée par la proposition 5.11. Les courbes obtenues sont parallèles, le comportement général de la probabilité est donc conservé par l'approximation. Ainsi, il est montré que la probabilité d'erreur diminue exponentiellement avec le nombre M de mots utilisés. De plus, pour obtenir une probabilité d'erreur inférieure à un certain seuil, le nombre de mots nécessaires n'augmente que de façon quadratique par rapport à la longueur de la permutation à reconstruire.

5.2.5 Résultats pratiques

Afin d'évaluer les performances de cette méthode de reconstruction de la permutation des turbo-codes, de nombreux tests ont été effectués. Cette sous-section est consacrée à la présentation des différents résultats et comportements observés lors des tests.

Trois modèles de canal de transmission sont considérés : le canal binaire à effacement, le canal binaire symétrique et le canal gaussien. Le calcul des probabilités $\boldsymbol{P}(u_t^s = a|\tilde{u}_t^s)$ et $\boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)$ est rappelé dans le chapitre 1 pour chacun de ces canaux (pour le canal gaussien, la modulation utilisée est la modulation BPSK dont les points sont d'abscisses -1 et 1).

Dans tous les tests présentés dans cette sous-section le codeur convolutif $\mathcal{C}_2$ utilisé est défini par $(1, \frac{1+D^2}{1+D+D^2})$.

Canal gaussien

Sur les figures 5.5 et 5.6 sont représentées les probabilités de ne pas reconstruire correctement la permutation en fonction du nombre de mots bruités utilisés, respectivement pour des longueurs de permutation de 64 et 512 et ceci pour différents niveaux de bruit (σ étant l'écart-type du canal gaussien considéré). Grâce à ces figures, il est mis en avant que plus le

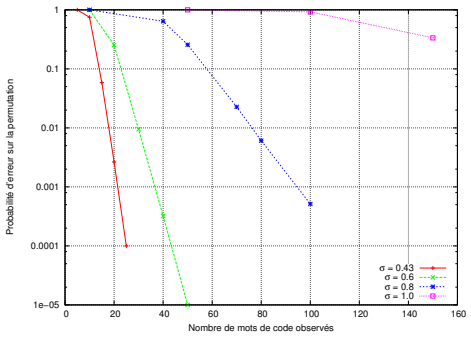


FIGURE 5.5 – Probabilité d'erreur sur la permutation, pour le canal gaussien et $N = 64$.

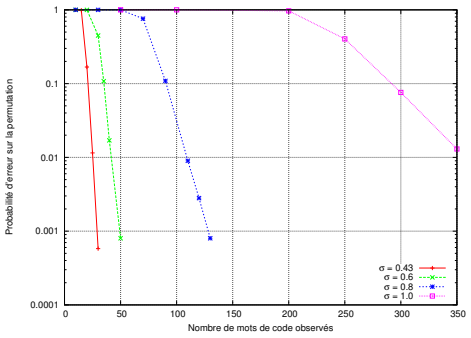


FIGURE 5.6 – Probabilité d'erreur sur la permutation, pour le canal gaussien et $N = 512$.

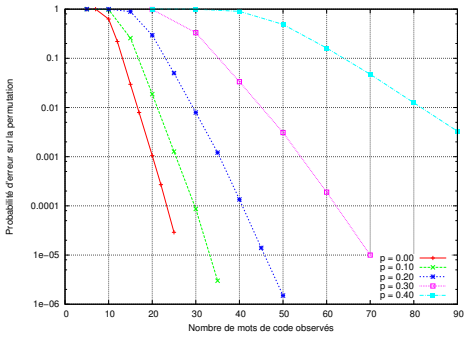


FIGURE 5.7 – Probabilité d'erreur sur la permutation, pour le canal à effacement et $N = 64$.

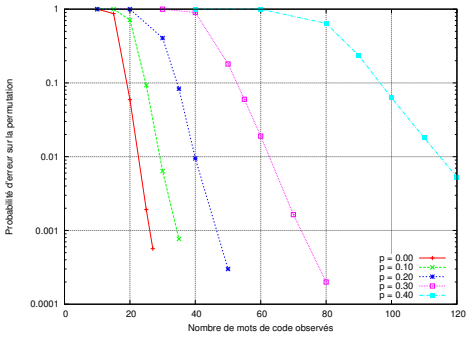


FIGURE 5.8 – Probabilité d'erreur sur la permutation, pour le canal à effacement et $N = 512$.

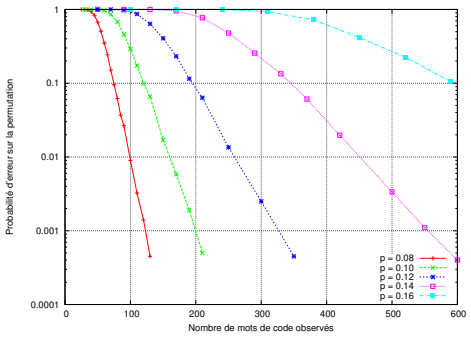


FIGURE 5.9 – Probabilité d'erreur sur la permutation, pour le canal binaire symétrique et $N = 64$.

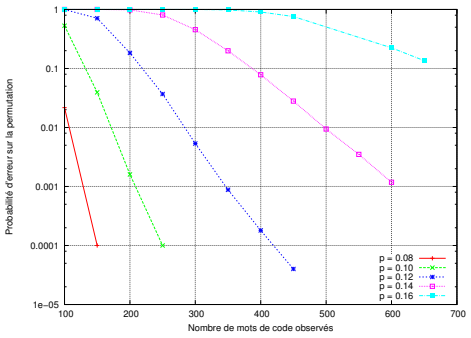


FIGURE 5.10 – Probabilité d'erreur sur la permutation, pour le canal binaire symétrique et $N = 100$.

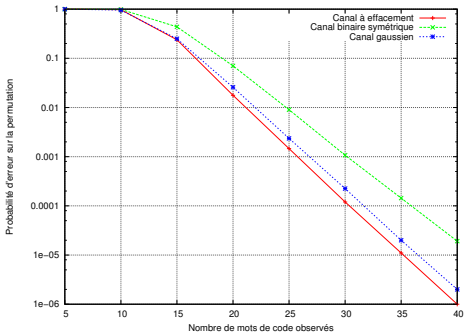


FIGURE 5.11 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.9, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1}{1+D})$.

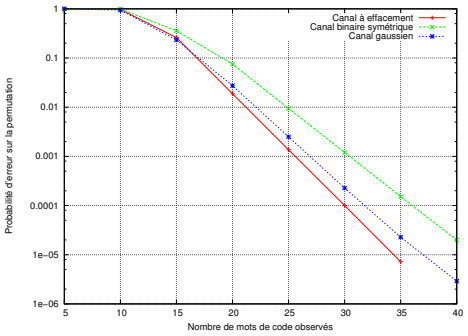


FIGURE 5.12 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.9, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1+D^2}{1+D+D^2})$.

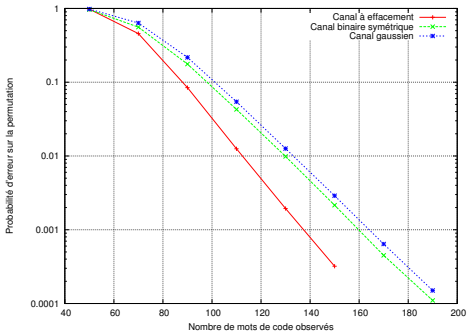


FIGURE 5.13 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.5, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1}{1+D})$.

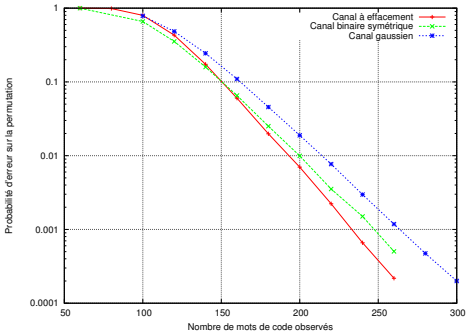


FIGURE 5.14 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.5, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1+D^2}{1+D+D^2})$.

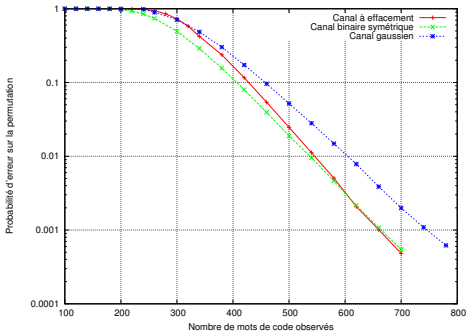


FIGURE 5.15 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.3, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1}{1+D})$.

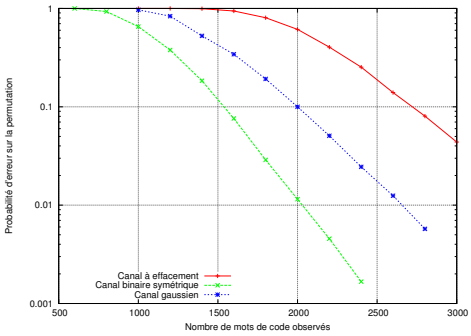


FIGURE 5.16 – Probabilité d'erreur sur la permutation pour une capacité de canal de 0.3, $N = 64$ et $\mathcal{C}_2 = (1, \frac{1+D^2}{1+D+D^2})$.

N	σ	M	M [25]	temps en secondes	temps en secondes [25]
64	0.43	20	50	0.003	0.2
64	0.6	34	115	0.005	0.3
64	1	243	1243	0.04	12
512	0.6	46	170	0.46	11
512	0.8	111	600	1.10	37
512	1	346	2 800	3.12	173
512	1.1	660	3 837	6.55	357
512	1.3	2 350	29 500	24	4 477
10 000	0.43	40	300	152	8 173

TABLE 5.1 – Pour le canal gaussien, nombre de mots nécessaire pour retrouver la permutation avec une probabilité d'erreur inférieure à 1%

niveau de bruit augmente plus le nombre de mots nécessaire est important, c'est également le cas en augmentant la longueur de la permutation. De plus, pour des niveaux de bruit relativement faibles, très peu de mots sont nécessaires pour reconstruire correctement la permutation. Par exemple, pour $N = 512$ et $\sigma = 0.6$, 50 mots sont suffisants pour retrouver la permutation dans plus 99.9% des cas. Enfin, sur ces figures, comme sur toutes les figures suivantes, le comportement exponentiel, en M, de la probabilité d'erreur est souligné par l'utilisation d'une échelle logarithmique sur l'axe des ordonnées.

Quant à la table 5.1 elle regroupe le nombre de mots nécessaire pour reconstruire correctement la permutation dans plus de 99% des cas ainsi que le temps pris par cette reconstruction. Ces valeurs sont comparées à celles obtenues par la méthode proposée par Cluzeau, Finiasz et Tillich [25]. Avec cette nouvelle méthode, le nombre de mots requis est nettement inférieur à celui de la méthode de référence [25], tout comme les temps de calcul. Cette table permet également de mettre en évidence le fait que la permutation est retrouvée rapidement même si le niveau de bruit est important (par exemple pour $\sigma = 1.3$) ou si sa longueur est très importante ($N \geq 10\,000$).

Canal à effacement

Les mêmes tests ont été effectués en simulant un canal à effacement, sur les figures 5.7 et 5.8 sont représentées l'évolution des probabilités d'erreur sur la permutation en fonction du nombre de mots observés, pour différents niveaux de bruits et en particulier dans le cas non bruité ($p = 0$) et pour des longueurs de permutation de $N = 64$ et $N = 512$.

La table 5.2 répertorie le nombre de mots nécessaire pour retrouver la permutation dans plus de 99% des cas, pour différentes probabilités d'effacement p et des longueurs de permutation comprises entre 32 et 256. Avec ce tableau, il peut être remarqué que la permutation peut être retrouvée, avec un nombre de mots relativement faible, même lorsque le niveau de bruit est supérieur à la capacité de correction du code convolutif (c'est par exemple le cas pour $p = 0.60$).

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	15	17	19	21
$p = 0.10$	19	22	24	27
$p = 0.20$	26	29	33	37
$p = 0.30$	37	45	50	58
$p = 0.40$	69	82	94	106
$p = 0.50$	160	193	225	
$p = 0.60$	540	663		860

TABLE 5.2 – Pour le canal à effacement, nombre de mots nécessaire pour retrouver la permutation avec une probabilité d'erreur inférieure à 1%.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	15	17	19	21
$p = 0.02$	26	30	34	37
$p = 0.04$	38	45	50	56
$p = 0.06$	55	66	75	84
$p = 0.08$	84	100	115	
$p = 0.10$	134	161	185	
$p = 0.12$	212	255		
$p = 0.14$	372		459	

TABLE 5.3 – Pour le canal binaire symétrique, nombre de mots nécessaire pour retrouver la permutation avec une probabilité d'erreur inférieure à 1%.

Canal binaire symétrique

De même pour le canal binaire symétrique de probabilité d'erreur p avec les figures 5.9 et 5.10 pour des permutations de longueurs $N = 64$ et $N = 100$. Pour ce modèle de canal, les seuils permettant de retrouver la permutation dans plus de 99% des cas sont donnés par la table 5.3. Dans cette table, la plus petite probabilité d'erreur considérée est $p = 0.02$ (hormis le cas sans erreur), ce niveau de bruit est, en pratique, un niveau élevé.

À capacité de canal fixée

Afin de mettre en évidence l'influence du modèle du canal utilisé, les figures 5.11 à 5.16 représentent la probabilité de ne pas reconstruire correctement la permutation en fonction du nombre de mots considéré, pour des canaux de capacité 0.9, 0.3 et 0.5 et ceci pour le codeur convolutif défini par $(1, \frac{1+D^2}{1+D+D^2})$ ainsi que pour celui défini par $(1, \frac{1}{1+D})$. Les performances de l'algorithme sont semblables quel que soit le canal utilisé. Cependant, à partir d'un certain nombre de mots, il est plus efficace sur le canal à effacement que sur les deux autres types de canaux. Les courbes de la figure 5.16 se croisant probablement pour un nombre de mots plus élevé, les temps de calculs importants n'ont pas permis de les prolonger suffisamment. Il est à noter que, comme le montre cette figure, la reconstruction peut être effectuée bien au delà de la capacité de correction du turbo-code.

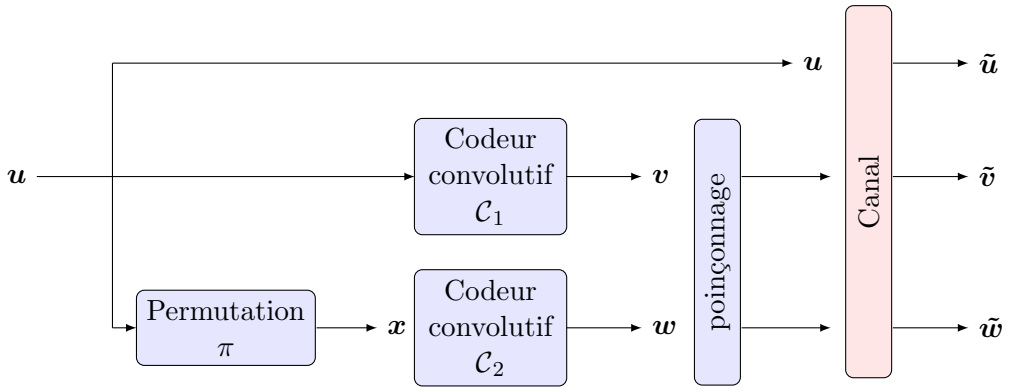


FIGURE 5.17 – Schéma d'un turbo-code parallèle poinçonné

5.3 Cas des turbo-codes parallèles poinçonnés

Les turbo-codes utilisés dans la section précédente sont de rendement 1/3 mais, comme expliqué dans le chapitre 2, ce rendement peut être augmenté en poinçonnant certains bits de redondance. La figure 5.17 reprend les différentes notations des turbo-codes parallèles en les appliquant aux turbo-codes parallèles poinçonnés, dans ce cas les vecteurs $\tilde{\boldsymbol{v}}$ et $\tilde{\boldsymbol{w}}$ ne sont pas de taille N mais de taille inférieure, celle-ci dépendant du nombre de bits poinçonnés.

Afin de pouvoir décoder les données reçues il est indispensable de connaître le motif du poinçonnage utilisé. Ce motif est donc supposé connu. Le poinçonnage le plus utilisé est celui consistant à ne transmettre qu'un bit sur deux des vecteurs de redondance $\boldsymbol{u}$ et $\boldsymbol{w}$. La méthode permettant de retrouver la permutation est ici adaptée pour ce poinçonnage, elle peut être très facilement adaptée aux autres motifs de poinçonnage. Cependant la complexité de l'algorithme dépend du nombre de bits consécutifs poinçonnés sur le vecteur $\boldsymbol{w}$.

Les mêmes notations que celles de la section précédente sont utilisées, en particulier :
— Si $\boldsymbol{a}$ est un vecteur alors la notation $\boldsymbol{a}_{i..j}$ représente le sous-vecteur $(a_i, \ldots, a_j)$
— Soit $\boldsymbol{a}^1, \ldots, \boldsymbol{a}^M$ une liste de vecteurs, la notation $\boldsymbol{a}_{i..j}^{k..l}$ représente l'ensemble $\{\boldsymbol{a}_{i..j}^k, \boldsymbol{a}_{i..j}^{k+1}, \ldots, \boldsymbol{a}_{i..j}^l\}$

De plus, pour se rapporter le plus possible du cas non poinçonné et faciliter les comparaisons, le vecteur $\tilde{\boldsymbol{w}}$ est artificiellement agrandi afin qu'il soit de taille N. Plus précisément, les bits poinçonnés sont replacés par des ?, comme s'il s'agissait de valeurs effacées lors de la transmission. Pour le motif de poinçonnage considéré $\tilde{\boldsymbol{w}} = (?, \tilde{w}_2, ?, \tilde{w}_4, \ldots, \tilde{w}_{N-2}, ?, \tilde{w}_N)$.

5.3.1 Principe de la méthode et calculs des probabilités

Le principe de l'algorithme est le même que pour le cas non poinçonné. Précédemment, à chaque instant t, l'algorithme recherche la valeur de j maximisant la probabilité :

$$\boldsymbol{P}(\pi(t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(t-1), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..t}^{1..M})$$

Pour cela la probabilité $\boldsymbol{P}(w_t = b | \tilde{w}_t)$ est utilisée or, dans le cas poinçonné, lorsque t est impair la valeur de $\tilde{w}_t$ est inconnue et ce pour l'ensemble des mots. Tous les indices j ont alors la même probabilité et l'algorithme ne peut en choisir un plus qu'un autre. Pour contrer

ce problème, la méthode est adaptée afin de retrouver simultanément $\pi(2t-1)$ et $\pi(2t)$ en utilisant la valeur de $\tilde{w}_{2t}$.

À chaque instant $t \in \{1, \ldots, N/2\}$, il est recherché le couple (i, j) maximisant la probabilité :

$$\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..2t}^{1..M})$$

De par l'indépendance des mots de code et de la même façon que précédemment cette probabilité s'exprime comme un produit :

$$\begin{aligned}&\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..2t}^{1..M}) \\ &= \rho \prod_{s=1}^{M} \boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{s}, \tilde{\boldsymbol{w}}_{1..2t}^{s})\end{aligned}$$

où ρ est un coefficient de normalisation tel que la somme des probabilités sur tous les couples possibles est égale à 1.

De manière similaire à la proposition 5.2, la proposition suivante permet de déterminer facilement chacun des termes du produit ci-dessus.

Proposition 5.12. *Soit e_t l'état interne du codeur $\mathcal{C}_2$ après que les t premiers bits d'information permutés sont entrés dans ce codeur. Soit la fonction :*

$$g(\alpha, a, c, d, i, j) \stackrel{def}{=} \boldsymbol{P}(\tilde{u}_i | u_i = a) \boldsymbol{P}(\tilde{u}_j | u_j = c) \boldsymbol{P}(\tilde{w}_{2t} | w_{2t} = d) \boldsymbol{P}(e_{2t-2} = \alpha | \tilde{\boldsymbol{u}}_{\tilde{\pi}(1)..\tilde{\pi}(2t-2)}, \tilde{\boldsymbol{w}}_{1..2t-2})$$

Alors :

$$\begin{aligned}&\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{s}, \tilde{\boldsymbol{w}}_{1..2t}^{s}) \\ &= \gamma \frac{1}{\boldsymbol{P}(\tilde{u}_i)\boldsymbol{P}(\tilde{u}_j)} \sum_{\alpha \in \{0,1\}^m} \sum_{(a,b,\eta):\alpha \stackrel{ab}{\to} \eta} \sum_{(c,d,\beta):\eta \stackrel{cd}{\to} \beta} g(\alpha, a, c, d, i, j)\end{aligned}$$

où γ est un coefficient de normalisation.

Démonstration. La preuve de cette proposition est similaire à celle de la proposition 5.2. □

Tout comme dans le cas non poinçonné, cette probabilité peut être exprimée en fonction des probabilités déduites des observations en sortie de canal : $\boldsymbol{P}(u_i = a|\tilde{u}_i)$, $\boldsymbol{P}(u_j = c|\tilde{u}_j)$ et $\boldsymbol{P}(w_{2t} = d|\tilde{w}_{2t})$ pour $a, c, d \in \{0, 1\}$.

5.3.2 Algorithme de reconstruction de la permutation

L'algorithme 4 découle des probabilités calculées précédemment, la complexité de cet algorithme est en $\mathcal{O}(2^m N^3 M)$.

L'annexe B.2 détaille sur un exemple le début de la reconstruction de la permutation d'un turbo-code parallèle poinçonné.

Algorithme 4 : RECONSTRUCTION_PERMUTATION_TC_PARALLÈLE_POINÇONNÉ.

Entrées :

1. Les probabilités $x_t^s(a) \stackrel{\text{def}}{=} \boldsymbol{P}(u_t^s = a|\tilde{u}_t^s)$ et $y_t^s(b) \stackrel{\text{def}}{=} \boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)$ pour tout $t \in \{1, \ldots, N\}$, tous $s \in \{1, \ldots, M\}$ et toutes les valeurs binaires a et b. //Pour t impair, $\boldsymbol{P}(w_t^s = b|\tilde{w}_t^s)$ n'est pas utilisé et correspond aux positions poinçonnées
2. Le deuxième codeur convolutif $\mathcal{C}_2$ de rendement $\frac{1}{2}$, et m sa mémoire.

Sortie : Une permutation $\tilde{\pi}$, de taille N, qui maximise, pour tout $t \in \{1, \ldots, N/2\}$, $\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j|\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..2t}^{1..M})$.

Initialisation des probabilités des états internes :
Pour $s = 1, \ldots, M$
  $p[s][0] \leftarrow 1$ //Les registres sont supposés initialisés à 0
  Pour $\alpha \in \{1, \ldots, 2^m - 1\}$
    $p[s][\alpha] \leftarrow 0$

Pour $t = 1, \ldots, N/2$
  Recherche des valeurs les plus probables de $\pi(2t-1)$ et $\pi(2t)$:
  Pour $s = 1, \ldots, M$
    Pour $i \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2)\}$
      Pour $j \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), i\}$
        $q[s][i][j] \leftarrow \sum_{\alpha \in \{0,1\}^m} \sum_{(a,b,\eta):\alpha \stackrel{ab}{\to} \eta} \sum_{(c,d,\beta):\eta \stackrel{cd}{\to} \beta} x_i^s(a) x_j^s(c) y_{2t}^s(d) p[s][\alpha]$
  Pour $i \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2)\}$
    Pour $j \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), i\}$
      $r[i][j] \leftarrow \prod_s q[s][i][j]$
  $\tilde{\pi}(2t-1) \leftarrow \arg_i \max_{i,j} r[i][j]$
  $\tilde{\pi}(2t) \leftarrow \arg_j \max_{i,j} r[i][j]$

  Mise à jour des probabilités des états internes :
  Pour $s = 1, \ldots, M$
    Pour $\beta \in \{0,1\}^m$
      $q[s][\beta] \leftarrow \sum_{\alpha:\alpha \stackrel{ab}{\to} \eta} \sum_{\eta:\eta \stackrel{cd}{\to} \beta} p[s][\alpha] x_{\tilde{\pi}(2t-1)}^s(a) x_{\tilde{\pi}(2t)}^s(c) y_{2t}^s(d)$
    Pour $\beta \in \{0,1\}^m$
      $p[s][\beta] \leftarrow \frac{q[s][\beta]}{\sum_\alpha q[s][\alpha]}$
Retourner $\tilde{\pi}$

5.3.3 Indéterminées et contre-mesures

En fonction du code convolutif $\mathcal{C}_2$ utilisé il peut arriver qu'à chaque instant t plusieurs couples (i, j) maximisent la probabilité :

$$\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..2t}^{1..M})$$

L'algorithme, tel que présenté ci-dessus, choisit alors, de façon aléatoire, l'un de ces couples. Si ce choix ne correspond pas aux bonnes valeurs, il peut être bloqué par la suite, c'est-à-dire qu'il existe $k > t$ tel que pour tous les couples (i, j) :

$$\boldsymbol{P}(\pi(2k-1) = i, \pi(2k) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2k-2), \tilde{\boldsymbol{u}}_{1..N}^{1..M}, \tilde{\boldsymbol{w}}_{1..2k}^{1..M}) = 0$$

Afin de contrer ce problème l'algorithme est légèrement adapté afin que la reconstruction se fasse en suivant un arbre de recherche. Autrement dit, à chaque instant t, il est recherché l'ensemble des couples (i, j) maximisant la probabilité souhaitée, l'un d'eux est choisi et s'il mène à une contradiction (c'est-à-dire que la reconstruction ne peut compléter entièrement la permutation), l'algorithme choisit le couple suivant et ainsi de suite.

En procédant de cette façon, la reconstruction est ralentie mais permet de trouver une ou des permutations compatibles avec les données observées. Le nombre de permutations retrouvées dépend du code convolutif utilisé, il est impossible de déterminer quelle est la permutation correcte parmi celles-ci puisque générant toutes les mêmes redondances. L'annexe C classe les codeurs convolutifs ayant une mémoire d'au plus 3 (comme ceux utilisés en pratique) en fonction du nombre de permutations retournées par l'algorithme lorsqu'un bit sur deux de $\boldsymbol{w}$ est poinçonné. Il peut être remarqué que très peu de codes convolutifs impliquent plus de 8 permutations possibles. De plus, ces permutations diffèrent, dans la plus part des cas, uniquement sur les tout derniers indices et n'impactent pas, ou très légèrement, le décodage.

5.3.4 Résultats pratiques

Comme pour le cas non poinçonné, de nombreux tests ont été effectués afin de déterminer les performances et les limites de cette méthode de reconstruction. La permutation est considérée comme correctement reconstruite si π appartient à la liste des permutations retournées par l'algorithme.

Pour ces tests le code convolutif $\mathcal{C}_2$ est défini par $(1, \frac{1+D^2}{1+D+D^2})$, il s'agit du même code que celui utilisé pour le cas non poinçonné. De plus, pour ce code et comme pour tous les codes tels que l'algorithme retourne seulement deux permutations, il est possible de l'adapter afin de le rendre plus rapide. En effet, pour ces codes, les valeurs de $\tilde{\pi}(2t-1)$ et $\tilde{\pi}(2t)$ sont fixées lors de la recherche de $\tilde{\pi}(2t+1)$ et $\tilde{\pi}(2t+2)$ il n'est donc pas nécessaire d'attendre que l'algorithme atteigne une contradiction pour déterminer les valeurs de $\tilde{\pi}(2t-1)$ et $\tilde{\pi}(2t)$.

La table 5.4 regroupe, pour deux longueurs de permutation et différents niveaux de bruit sur le canal gaussien les nombres de mots nécessaires afin de retrouver la permutation dans plus de 99% des cas ainsi que les temps de calcul correspondant. L'amélioration précédente est utilisée dans ces tests mais malgré cela les temps de calculs restent assez élevés comparés à ceux du cas non poinçonné de la table 5.1.

N	σ	M	temps en secondes
64	0.43	25	0.18
64	0.6	56	0.41
64	1	1410	10.4
512	0.6	82	330

TABLE 5.4 – Turbo-code parallèle poinçonné, canal gaussien, nombre de mots nécessaire pour reconstruire la permutation avec une probabilité supérieure à 99%.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	18	22	25	28
$p = 0.10$	27	32	36	42
$p = 0.20$	45	55	63	71
$p = 0.30$	95	118		
$p = 0.40$	267	346		

TABLE 5.5 – Turbo-code parallèle poinçonné, canal à effacement, nombre de mots nécessaire pour retrouver la permutation avec une probabilité supérieure à 99%.

Les tables 5.5 et 5.6 rassemblent les nombres de mots nécessaires pour reconstruire la permutation dans plus de 99% des cas respectivement pour le canal à effacement et le canal binaire symétrique. Ces tables peuvent être comparées aux tables 5.2 et 5.3 qui représentent les mêmes valeurs mais pour le cas non poinçonné.

Enfin, les figures 5.18 à 5.20 représentent l'évolution de la probabilité de ne pas reconstruire correctement la permutation en fonction du nombre de mots bruités utilisés et ceci respectivement pour le canal gaussien, le canal à effacement et le canal binaire symétrique.

5.4 Cas des turbo-codes série

Dans cette section l'algorithme de reconstruction de la permutation est adapté aux cas des turbo-codes série. Les notations utilisées pour ce type de turbo-code sont regroupées sur la figure 5.21. Afin que, comme pour les autres types de turbo-codes, la permutation soit de taille N, le vecteur $\boldsymbol{u}$ est de taille $N/2$.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	18	22	25	28
$p = 0.02$	39	47	55	63
$p = 0.04$	69	85	100	
$p = 0.06$	123	154	186	
$p = 0.08$	226	293		
$p = 0.10$	454			
$p = 0.12$	903			

TABLE 5.6 – Turbo-code parallèle poinçonné, canal binaire symétrique, nombre de mots nécessaire pour retrouver la permutation avec une probabilité supérieure à 99%.

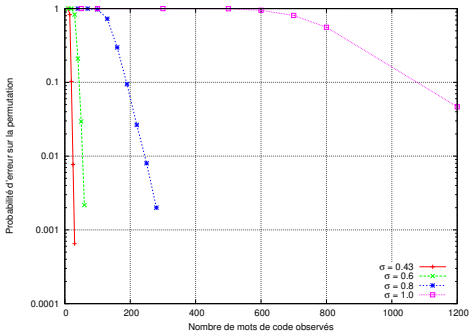


FIGURE 5.18 – Turbo-code parallèle poinçonné, probabilité d'erreur sur la permutation, pour le canal gaussien, $N = 64$.

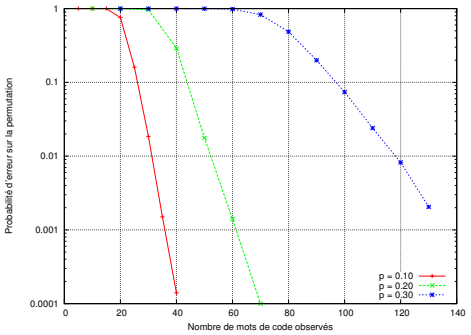


FIGURE 5.19 – Turbo-code parallèle poinçonné, probabilité d'erreur sur la permutation, pour le canal à effacement, $N = 64$.

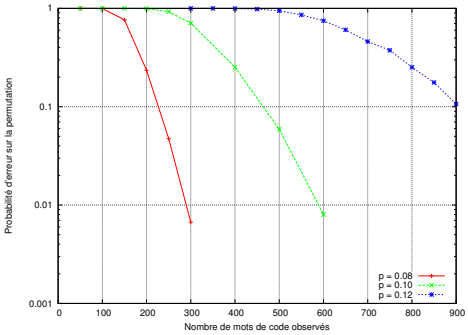


FIGURE 5.20 – Turbo-code parallèle poinçonné, probabilité d'erreur sur la permutation, pour le canal binaire symétrique, $N = 64$.

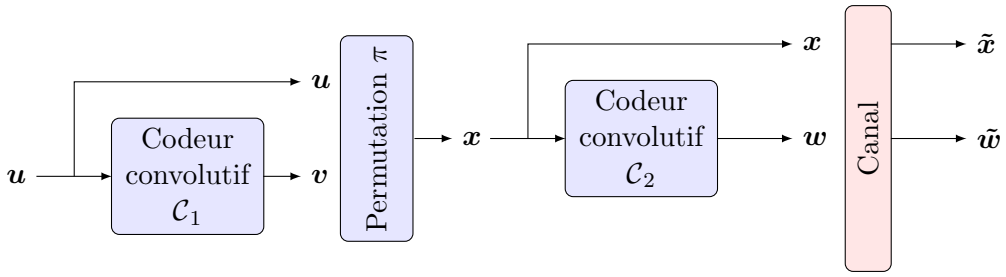


FIGURE 5.21 – Notations pour les turbo-codes série.

5.4.1 Principe de la méthode et calculs des probabilités

Pour la reconstruction de la permutation de ce type de turbo-code les hypothèses sont semblables à celles des turbo-codes parallèles mais portent sur le premier code convolutif. En effet, dans le cas parallèle le code $\mathcal{C}_1$ peut être retrouvé grâce aux méthodes classiques de reconnaissance des codes convolutifs. Alors que pour les turbo-codes série, c'est le code $\mathcal{C}_2$ qui est retrouvé en utilisant les sorties $\tilde{\boldsymbol{x}}$ et $\tilde{\boldsymbol{w}}$, les hypothèses portent donc sur le code $\mathcal{C}_1$. Ce dernier est donc supposé connu, systématique, de dimension 1 et de longueur 2. Comme précédemment, il est tel que le premier bit de sortie dépend du premier bit entré dans le codeur et les registres sont réinitialisés à 0 au début du codage de chacun des mots.

Soit $\boldsymbol{a} = (u_1, v_1, u_2, v_2, \ldots, u_{N/2}, v_{N/2})$ l'entrée de la permutation, ce vecteur correspond à la sortie du codeur convolutif $\mathcal{C}_1$ en alternant un bit systématique avec un bit de redondance. Par construction, $\pi(\boldsymbol{a}) = \boldsymbol{x}$. L'algorithme de reconstruction de la permutation cherche donc à retrouver ce vecteur $\boldsymbol{a}$ en utilisant uniquement le vecteur $\tilde{\boldsymbol{x}}$ et les hypothèses faites sur le code $\mathcal{C}_1$. Cette reconstruction s'effectue pas à pas et retrouve à chaque instant t les valeurs du couple $(\tilde{\pi}(2t-1), \tilde{\pi}(2t))$. Pour cela, la probabilité

$$\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{x}}_{1..N}^{1..M})$$

est déterminée pour tous les couples (i, j). De façon similaire aux cas parallèles, cette probabilité se décompose en un produit de probabilités grâce à l'indépendance des mots de code. Toujours de la même façon, elle s'exprime en fonction des probabilités déduites des observations faites en sortie de canal.

Comme pour le cas des turbo-codes parallèles poinçonnés, il y a de nombreuses indéterminées. En effet, à chaque instant t, l'égalité suivante est satisfaite :

$$\begin{aligned}&\boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{x}}_{1..N}^{1..M}) \\ &= \boldsymbol{P}(\pi(2t-1) = j, \pi(2t) = i | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{x}}_{1..N}^{1..M})\end{aligned}$$

Cependant il n'est pas nécessaire que l'algorithme suive un arbre de recherche, puisque ces indéterminées ne dépendent pas du code utilisé et sont levées dès l'instant t suivant. Deux permutations possèdent donc la même probabilité d'être celle utilisée et ne peuvent être différenciées, ces deux permutations diffèrent uniquement sur les deux derniers indices.

Une autre de ces indéterminées peut, selon les cas, ne pas être levée, elle concerne les deux premiers indices de la permutation. En effet, en supposant que les registres du codeur sont initialisés à 0, il peut arriver, à l'instant $t = 1$, que la redondance soit égale à la valeur entrée dans le codeur. Dans ce cas il est impossible de déterminer quelle est la valeur correspondant à l'information par rapport à celle de redondance, puisqu'elles sont égales. Cette indéterminée, comme celle sur les deux derniers indices, n'a aucun impact sur le décodage des mots bruités observés.

5.4.2 Algorithme de reconstruction de la permutation

L'algorithme 7 recherche la permutation la plus probablement utilisée dans un turbo-code série à partir d'un ensemble de mots de code bruités. Celui-ci fait appel aux algorithmes intermédiaires 5 et 6.

Algorithme 5 : PROBABILITÉS_DES_ÉTATS.

Entrées :

1. Les mêmes que l'algorithme 7
2. t l'instant considéré
3. $p[.][.]$ les probabilités des états à l'instant $2t-2$
4. $(\tilde{\pi}(2t-1), \tilde{\pi}(2t))$ les nouveaux indices de la permutation

Sortie : $p'[.][.]$ les probabilités des états à l'instant $2t$

Pour $s \in \{1, \ldots, M\}$
  Pour $\beta \in \{0,1\}^m$
    $q[s][\beta] \leftarrow \sum_{\alpha:\alpha \overset{ab}{\to} \beta} \boldsymbol{P}(x^s_{\tilde{\pi}(2t-1)} = a | \tilde{x}^s_{\tilde{\pi}(2t-1)}) \boldsymbol{P}(x^s_{\tilde{\pi}(2t)} = b | \tilde{x}^s_{\tilde{\pi}(2t)}) p[s][\alpha]$
  Pour $\beta \in \{0,1\}^m$
    $p'[s][\beta] \leftarrow \frac{q[s][\beta]}{\sum_{\alpha \in \{0,1\}^m} q[s][\alpha]}$
Retourner $p'[.][.]$

Algorithme 6 : RECHERCHE_COUPLE.

Entrées :

1. Les mêmes que l'algorithme 7
2. t l'instant considéré
3. $p[.][.]$ les probabilités des états à l'instant $2t-2$

Sortie : $(i, j, prob)$ le couple (i, j) le plus probable pour $\pi(2t-1), \pi(2t)$ et sa probabilité : $prob = \boldsymbol{P}(\pi(2t-1) = i, \pi(2t) = j | \tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), \tilde{\boldsymbol{x}}^{1..M}_{1..N})$

Pour $i, j \in \{1, \ldots, N\}$
  $r[i][j] \leftarrow -1$
Pour $s \in \{1, \ldots, M\}$
  Pour $i \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2)\}$
    Pour $j \in \{1, \ldots, N\} \setminus \{\tilde{\pi}(1), \ldots, \tilde{\pi}(2t-2), i\}$
      Si $r[i][j] = -1$
        $r[i][j] \leftarrow 1$
      $z \leftarrow \sum_{\alpha \in \{0,1\}^m} \sum_{\beta:\alpha \overset{ab}{\to} \beta} \boldsymbol{P}(x^s_i = a | \tilde{x}^s_i) \boldsymbol{P}(x^s_j = b | \tilde{x}^s_j) p[s][\alpha]$
      $r[i][j] \leftarrow r[i][j] \times z$
$i \leftarrow arg_a \max_{a,b} r[a][b]$
$j \leftarrow arg_b \max_{a,b} r[a][b]$
$prob \leftarrow r[i][j]$
Retourner $(i, j, prob)$

Algorithme 7 : RECONSTRUCTION_PERMUTATION_TURBO-CODE_SÉRIE.

Entrées :

1. Les probabilités $\boldsymbol{P}(x_t^s = a|\tilde{x}_t^s)$ pour tout $t \in \{1, \ldots, N\}$, tous $s \in \{1, \ldots, M\}$ et $a \in \{0, 1\}$.
2. Le premier codeur convolutif $\mathcal{C}_1$ de rendement $\frac{1}{2}$, et m sa mémoire.

Sortie : $\tilde{\pi}$ la permutation la plus probablement utilisée

Initialisation des probabilités des états internes :
Pour $s = 1, \ldots, M$
  $p[s][0] \leftarrow 1$ //Les registres sont supposés initialisés à 0
  Pour $\alpha \in \{1, \ldots, 2^m - 1\}$
    $p[s][\alpha] \leftarrow 0$

$(a, b, prob_1) \leftarrow$RECHERCHE_COUPLE$(0, p[.][.])$
Pour $t \in \{2, \ldots, N/2\}$
  //Si $\tilde{\pi}(2t-3) = a$ et $\tilde{\pi}(2t-2) = b$:
  $p_{etat_1} \leftarrow$ PROBABILITÉS_DES_ÉTATS$(t, p[.][.], a, b)$
  $(a_1, b_1, prob_1) \leftarrow$ RECHERCHE_COUPLE$(t, p_{etat_1}[.][.])$
  //Si $\tilde{\pi}(2t-3) = b$ et $\tilde{\pi}(2t-2) = a$:
  $p_{etat_2} \leftarrow$ PROBABILITÉS_DES_ÉTATS$(t, p[.][.], b, a)$
  $(a_2, b_2, prob_2) \leftarrow$ RECHERCHE_COUPLE$(t, p_{etat_2}[.][.])$
  Si $prob_1 > prob_2$
    $\tilde{\pi}(2t-3) \leftarrow a$
    $\tilde{\pi}(2t-2) \leftarrow b$
    $a \leftarrow a_1$
    $b \leftarrow b_1$
    $p[.][.] \leftarrow p_{etat_1}[.][.]$
  Sinon
    $\tilde{\pi}(2t-3) \leftarrow b$
    $\tilde{\pi}(2t-2) \leftarrow a$
    $a \leftarrow a_2$
    $b \leftarrow b_2$
    $p[.][.] \leftarrow p_{etat_2}[.][.]$
  Retourner $\tilde{\pi}$

N	σ	M	temps en secondes
64	0.43	23	0.17
64	0.6	39	0.30
64	1	305	2.59
512	0.6	57	221

TABLE 5.7 – Turbo-code série, nombre de mots nécessaire pour reconstruire la permutation avec une probabilité supérieure à 99%.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	18	22	25	28
$p = 0.10$	23	27	31	35
$p = 0.20$	31	36	42	48
$p = 0.30$	45	55	62	73
$p = 0.40$	77	97	115	
$p = 0.50$	167	222		

TABLE 5.8 – Turbo-code série, canal à effacement, nombre de mots nécessaire pour retrouver la permutation avec une probabilité supérieure à 99%.

5.4.3 Résultats pratiques

Comme précédemment, la méthode de reconstruction de la permutation a été testée sur de nombreux exemples. Pour les résultats présentés dans cette sous-section le code convolutif $\mathcal{C}_1$ est défini par $(1, \frac{1+D^2}{1+D+D^2})$.

La table 5.7 indique le nombre de mots nécessaire pour retrouver la permutation dans plus de 99% des cas en fonction de la longueur de la permutation et de l'écart type σ du canal gaussien utilisé. Cette table peut être comparée aux tables 5.1 et 5.4. Cette comparaison met en évidence que même si la reconnaissance se fait paire par paire comme pour les turbo-codes parallèles poinçonnés, elle nécessite une quantité de donnée inférieure (proche du cas des turbo-codes parallèles).

Les tables 5.8 et 5.9 indiquent elles-aussi les nombres de mots utiles pour retrouver la permutation dans plus de 99% des cas mais pour les canaux à effacement et binaire symétrique.

De la même façon que pour les turbo-codes parallèles, poinçonnés ou non, les figures 5.22 à 5.24 représentent l'évolution de la probabilité de ne pas retrouver la permutation en fonction du nombre de mots observés et ceci respectivement pour les canaux gaussien, à effacement et binaire symétrique.

	$N = 32$	$N = 64$	$N = 128$	$N = 256$
$p = 0$	18	22	25	28
$p = 0.02$	31	38	43	50
$p = 0.04$	46	56	66	74
$p = 0.06$	66	83	100	
$p = 0.08$	97	122	148	
$p = 0.10$	147	192		
$p = 0.12$	233	313		

TABLE 5.9 – Turbo-code série, canal binaire symétrique, nombre de mots nécessaire pour retrouver la permutation avec une probabilité supérieure à 99%.

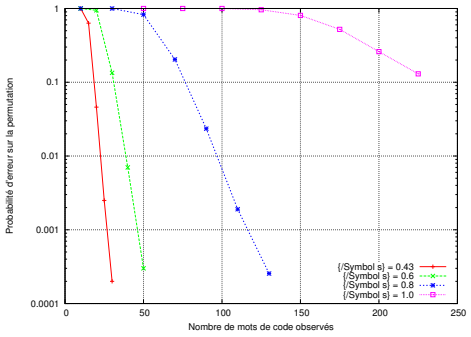


FIGURE 5.22 – Turbo-code série, probabilité d'erreur sur la permutation pour le canal gaussien et $N = 64$

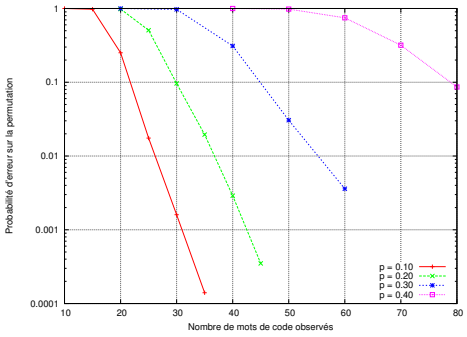


FIGURE 5.23 – Turbo-code série, probabilité d'erreur sur la permutation pour le canal à effacement et $N = 64$

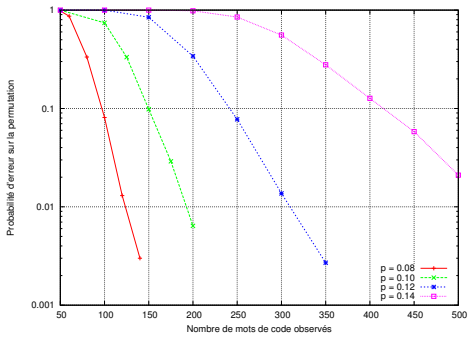


FIGURE 5.24 – Turbo-code série, probabilité d'erreur sur la permutation pour le canal binaire symétrique et $N = 64$

5.5 Conclusion

Les méthodes présentées permettent de retrouver efficacement la permutation utilisée quel que soit le type du turbo-code. Cependant, c'est pour les turbo-codes parallèles non-poinçonnés que les résultats sont les plus pertinents de par la rapidité de la reconstruction mais également grâce à la quantité de données nécessaire qui est très faible comparée aux méthodes existantes. Ces deux comportements ont été mis en évidence lors des différents tests, tout comme l'efficacité de cet algorithme sur des tailles de permutation très importantes. De plus, la permutation est reconstruite même lorsque le niveau de bruit dépasse largement la capacité de correction du code convolutif, pour cela il suffit d'avoir suffisamment de données.

Le poinçonnage ralentit l'algorithme par la présence d'indéterminées qui impliquent de nombreux retours en arrière lors de la reconstruction. Même si ces indéterminées ne peuvent pas toutes être levées, le nombre de permutations retournées par l'algorithme est très faible (inférieur à 10 dans l'immense majorité des cas) et ne dépend pas de la longueur N de la permutation. De plus, choisir l'une de ces permutations plutôt qu'une autre impacte uniquement les tout derniers bits d'information lors du décodage.
Comme expliqué précédemment, l'algorithme de reconstruction de la permutation des turbo-code parallèles poinçonnés est adapté ici à un motif de poinçonnage donné, mais il est facilement ajustable aux autres motifs. La complexité de l'algorithme dépend alors du nombre de bits poinçonnés consécutivement dans la redondance émise par le second codeur convolutif.

Concernant les turbo-codes série, la quantité de données nécessaire reste relativement faible par rapport au niveau de bruit introduit par le canal et à la longueur de la permutation. Les indéterminées intermédiaires sont très rapidement levées, l'algorithme en est quand même ralenti. La méthode proposée atteint donc ses limites avec des longueurs de permutations élevées. Or, pour ce type de turbo-code, le problème est exactement le même que celui de la reconnaissance d'un code convolutif entrelacé. Le chapitre 8 est consacré à la résolution de ce problème, la méthode qui y est proposée peut donc être appliquée aux turbo-codes série, cependant pour des cas très bruités, l'algorithme qui vient d'être décrit ici est plus approprié.

Chapitre 6

Reconnaissance des codes LDPC : Recherche des équations de parité de petits poids

Pour reconstruire un code linéaire $\mathcal{C}$ de longueur N à partir d'un ensemble de M mots de code bruités $\tilde{\boldsymbol{m}}^1, \tilde{\boldsymbol{m}}^2, \ldots, \tilde{\boldsymbol{m}}^M$, plusieurs méthodes existent. Ces méthodes ont presque toutes le même objectif : retrouver le code dual. Pour cela, elles recherchent les équations de parité vérifiées par les mots de code. Sans hypothèse supplémentaire, la recherche du code dual est un problème NP-complet (démontré par Valembois en 2001 [130]). Cependant, si le poids des équations de parité est constant le coût de leur recherche est polynomial, par contre l'exposant de ce coût est très élevé. L'objectif est donc de retrouver les équations de parité de poids constant tout en minimisant le coût de la recherche. Cette hypothèse sur le poids constant des équations de parité s'applique notamment aux codes convolutifs, qui, par construction, possèdent de nombreuses équations de (petit) poids constant ainsi qu'aux codes LDPC et en particulier aux codes LDPC réguliers.

Pour reconnaître $\mathcal{C}$, un code LDPC, une solution consiste à reconstruire une de ses matrices de parité. Par définition des codes LDPC, il existe une matrice de parité $\boldsymbol{\mathcal{H}}$ très creuse, c'est cette matrice qui est recherchée. En effet, de par les petits poids de ses lignes c'est la matrice la plus facilement retrouvable. Ceci est dû à la probabilité qu'un mot appartenant au code $\mathcal{C}$, bruité par un canal binaire symétrique de probabilité d'erreur τ, satisfasse une équation de parité de poids t du code $\mathcal{C}$ qui est égale à $\frac{1+(1-2\tau)^t}{2}$. En d'autres termes, le biais (par rapport à $1/2$) qu'une équation de parité soit satisfaite diminue exponentiellement avec t, et l'on s'attend logiquement à avoir beaucoup plus de difficultés pour retrouver des équations de parité de poids élevés.

La reconnaissance d'un code LDPC repose donc sur la recherche des équations de parité de petit poids. Comme dit précédemment, cette recherche d'équations n'est pas uniquement adaptée aux codes LDPC. C'est une méthode de base très largement utilisée, elle sert, entre autre, pour :

— la reconnaissance des codes convolutifs,
— la reconnaissance des codes convolutifs entrelacés,
— la reconnaissance des turbo-codes.

En effet, ces familles de codes satisfont de nombreuses équations de parité de petit poids,

retrouver la structure liant ces équations est une solution pour les reconnaître, d'où l'intérêt de trouver efficacement des équations de parité vérifiées par les mots de code.
Il peut être noté que la recherche d'équations de petit poids est aussi utilisée en cryptanalyse [80, 81] notamment sur le système de McEliece [106].

Contribution. Dans ce chapitre, il est présenté une nouvelle méthode pour retrouver un ensemble d'équations de parité de petits poids. Cette méthode, basée sur l'algorithme de Dumer, généralise les méthodes existantes et permet de retrouver plus efficacement un ensemble d'équations de parité de petits poids. Les équations retrouvées servent dans les chapitres suivants pour reconstruire les codes LDPC ainsi que les codes convolutifs entrelacés.

Hypothèses. Les hypothèses qui sont faites dans la suite de ce chapitre sont :
— N la longueur du code $\mathcal{C}$ est connue ;
— τ la probabilité d'erreur du canal binaire symétrique utilisé est connue ;
— M mots de code bruités $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ sont donnés. Les mots de code (non bruités) correspondant sont notés $\boldsymbol{m}^1, \ldots, \boldsymbol{m}^M$, ils sont évidemment inconnus.

La dimension K du code $\mathcal{C}$ n'est pas nécessairement connue, cependant il est fréquent de connaître K ou au moins une estimation de sa valeur. La connaissance de K permet de déterminer plus facilement si la recherche des équations de parité est terminée, mais elle n'est pas indispensable.

6.1 Méthodes existantes

6.1.1 Méthode de Gauss Randomisé

La méthode de recherche des équations de parité la plus simple est celle dite de Gauss. Elle est particulièrement bien adaptée au cas non bruité et utilise le fait qu'un mot $\boldsymbol{h} = (h_1, \ldots, h_N)$ appartenant au code dual de $\mathcal{C}$ vérifie le système :

$$\begin{cases} \boldsymbol{m}^1 \boldsymbol{h}^T = \boldsymbol{0} \\ \boldsymbol{m}^2 \boldsymbol{h}^T = \boldsymbol{0} \\ \ldots \\ \boldsymbol{m}^M \boldsymbol{h}^T = \boldsymbol{0} \end{cases}$$

Notons $\boldsymbol{\mathcal{M}}$ la matrice obtenue à partir des données observées :

$$\boldsymbol{\mathcal{M}} = \begin{pmatrix} \tilde{\boldsymbol{m}}^1 \\ \tilde{\boldsymbol{m}}^2 \\ \vdots \\ \tilde{\boldsymbol{m}}^M \end{pmatrix}$$

Si les données $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ ne sont pas bruitées, la matrice $\boldsymbol{\mathcal{M}}$ est de rang K, des solutions du système précédent s'obtiennent alors en effectuant un pivot de Gauss sur $\boldsymbol{\mathcal{M}}$.
Dans le cas bruité, on essaiera de tirer parti de cette propriété en utilisant le fait que cette matrice est (assez souvent) de rang non plein quand un mot du code dual est orthogonal à l'ensemble des mots bruités contenus dans la matrice. Pour retrouver l'ensemble des équations de parité, la méthode est itérée en partant de différentes matrices $\boldsymbol{\mathcal{M}}$.

Le principe de la méthode proposée par Sicot et Houcke [118–120], dite méthode de Gauss Randomisé, est le même à la différence près qu'il est recherché des mots orthogonaux aux N premiers mots, les $M - N$ autres servant uniquement à valider (ou non) le fait que les mots retrouvés appartiennent au code dual. Avec cette méthode il est donc nécessaire de disposer de plus de N mots.

Coût de la méthode de Gauss Randomisé

Dans l'article [121], les auteurs donnent le nombre moyen d'équations de parité retrouvées lors d'une itération. Ce nombre est :

$$(N-K)\left(\frac{1+(1-2\tau)^t}{2}\right)^N,$$

où la partie fractionnaire est égale à la probabilité que les N mots $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ satisfassent une équation de parité de poids t du code $\mathcal{C}$.

La dépendance exponentielle en N explique que cette méthode n'est vraiment adaptée qu'à des niveaux de bruit très faibles et/ou à des codes de faibles longueurs.

Le coût d'une itération est dominé par le pivot de Gauss, il possède un coût cubique en N. De ce coût et du nombre moyen d'équations retrouvées en une itération il se déduit le coût moyen C_{GR} pour retrouver une équation de parité par la méthode de Gauss Randomisé :

$$C_{GR} = \frac{N^3}{(N-K)\left(\frac{1+(1-2\tau)^t}{2}\right)^N}$$

Cette valeur est déterminée pour différentes méthodes afin de les comparer, quelques exemples sont regroupés dans la table 6.3 de la section 6.5.

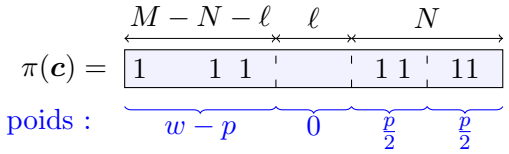
6.1.2 Méthode de Valembois

Dans cette méthode [130], comme dans toutes les suivantes, la recherche d'équations de petit poids est ramenée à la recherche de mots de poids faible dans le code généré par la matrice $\mathcal{X}$ suivante :

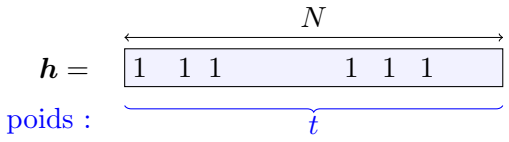
$$\mathcal{X} = \begin{pmatrix} \tilde{\boldsymbol{m}}^{1^T} & \tilde{\boldsymbol{m}}^{2^T} & \cdots & \tilde{\boldsymbol{m}}^{M^T} \end{pmatrix}$$

Effectivement, le but étant de trouver des équations de parité de poids t, la méthode de Valembois recherche des vecteurs $\boldsymbol{h}$ de poids t tels que le vecteur $\boldsymbol{c}$ défini par $\boldsymbol{c} = \boldsymbol{h}\mathcal{X}$ soit de petit poids. Ceci s'explique par le fait que le poids de $\boldsymbol{c}$, noté w, est égal au nombre de mots observés ne vérifiant pas l'équation de parité $\boldsymbol{h}$. La valeur de w peut alors être estimée à priori : $w \simeq \frac{1-(1-2\tau)^t}{2} M$. t étant petit, le poids w de $\boldsymbol{c}$ est également petit. De plus, par construction, $\boldsymbol{c}$ est un mot appartenant au code généré par la matrice $\mathcal{X}$ d'où la méthode de Valembois qui consiste à rechercher un mot de petit poids dans ce code.

Pour cela, Valembois utilise la méthode de Stern [122] (méthode que l'on détaillera un peu plus loin). Il s'agit d'une méthode itérative, au début de chaque itération une permutation π est choisie, les mots $\boldsymbol{c}$ retrouvés sont tels que $\pi(\boldsymbol{c})$ possède la répartition de poids :



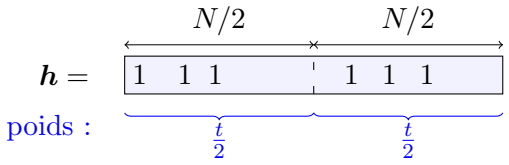
quant à l'équation de parité $\boldsymbol{h}$ qui s'en déduit, sa répartition de poids est donnée par :



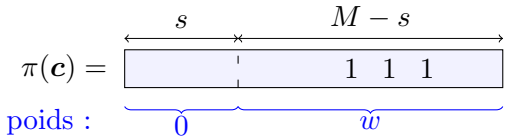
Il peut être remarqué qu'il est nécessaire de disposer de plus de N mots pour appliquer cette méthode. De plus, une des étapes de cette méthode est un pivot de Gauss, de manière à modérer son impact sur le coût de la recherche, Valembois conseille l'utilisation de la méthode de Canteaut-Chabaud [14].

6.1.3 Méthode de Cluzeau-Finiasz

La méthode donnée par Cluzeau et Finiasz [24] utilise le paradoxe des anniversaires afin de retrouver des équations de parité $\boldsymbol{h}$ dont la répartition de poids est :



Les mots de code $\boldsymbol{c}$ de petit poids qui sont retrouvés lors d'une itération sont tels que $\pi(\boldsymbol{c})$ possède la répartition de poids :



Plus précisément, la méthode consiste à partager horizontalement la matrice $\mathcal{X}$ en deux sous-matrices $\mathcal{X}_1$ et $\mathcal{X}_2$ de même taille. Afin de retrouver toutes les équations de parité ayant la répartition de poids donnée ci-dessus, deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$ sont crées. La liste $\mathcal{L}_i$ contient les s premiers indices de tous les produits d'un vecteur de poids $t/2$ par la matrice $\mathcal{X}_i$. Une recherche de collisions entre ces deux listes permet la déduction d'un ensemble de vecteurs $\boldsymbol{h}$ de poids t tels que $\boldsymbol{h}\mathcal{X}$ est nul sur ses s premiers indices. Il faut ensuite vérifier le poids de ce vecteur sur ses autres indices afin d'en déduire si $\boldsymbol{h}$ est l'une des équations de parité recherchées.

Coût de la méthode de Cluzeau-Finiasz

Afin qu'une équation de parité donnée soit retrouvée, il est nécessaire que les s premiers mots ne possèdent pas d'erreur ou un nombre pair d'erreurs sur le support de l'équation

recherchée. Ceci se produit avec une probabilité égale à (formule classique des codes LDPC, elle est donnée dans [24])

$$\left(\frac{1+(1-2\tau)^t}{2}\right)^s$$

La probabilité qu'une équation de poids t possède la répartition de poids donnée ci-dessus est :

$$\frac{\binom{N/2}{t/2}^2}{\binom{N}{t}}$$

En supposant que le code recherché possède $N-K$ équations de parité de poids t, le nombre moyen d'équations de parité retrouvées lors d'une itération est donné par :

$$(N-K)\frac{\binom{N/2}{t/2}^2}{\binom{N}{t}}\left(\frac{1+(1-2\tau)^t}{2}\right)^s$$

Le coût d'une itération peut être approximé par la formule suivante, il correspond au coût de la fabrication de deux listes de tailles $\binom{N/2}{t/2}$ ainsi qu'au coût de la recherche de leur intersection.

$$2\binom{N/2}{t/2}+2\frac{\binom{N/2}{t/2}^2}{2^s}$$

D'où le coût moyen C_{CF} pour retrouver une équation de parité de poids t par la méthode de Cluzeau-Finiasz :

$$C_{CF}=\frac{2\binom{N/2}{t/2}+2\frac{\binom{N/2}{t/2}^2}{2^s}}{(N-K)\binom{N/2}{t/2}^2\left(\frac{1+(1-2\tau)^t}{2}\right)^s}\binom{N}{t}$$

Pour rappel, les coûts des différentes méthodes sont comparés dans la section 6.5 et plus précisément grâce à la table 6.3.

6.1.4 Méthodes basées sur l'Information Set Decoding (ISD)

La recherche d'équations de parité de petit poids se ramène à la recherche de mots de petits poids dans un code, par exemple le code de matrice génératrice $\mathcal{X}$. Les méthodes de décodage des codes en bloc de type Information Set Decoding (ISD) peuvent alors être utilisées.

Plus précisément, les algorithmes de type ISD prennent en entrée un mot de code bruité $\tilde{\boldsymbol{m}}$ et recherchent le vecteur d'erreur $\boldsymbol{e}$ (de petit poids) afin d'en déduire le mot de code (non bruité) $\boldsymbol{m}$ appartenant au code $\mathcal{C}$ tel que $\tilde{\boldsymbol{m}}=\boldsymbol{m}+\boldsymbol{e}$.

Soit $\boldsymbol{\mathcal{H}}$ une matrice de parité du code $\mathcal{C}$, de longueur N et de dimension K. Par définition, $\tilde{\boldsymbol{m}}$ peut se décomposer en la somme d'un mot de code $\boldsymbol{m}=(m_1,\ldots,m_N)$ et d'un vecteur d'erreur $\boldsymbol{e}=(e_1,\ldots,e_N)$. Notons $\boldsymbol{s}$ le syndrome associé à $\tilde{\boldsymbol{m}}$: $\boldsymbol{s}^T=\boldsymbol{\mathcal{H}}\tilde{\boldsymbol{m}}^T$. Par construction, $\boldsymbol{s}^T=\boldsymbol{\mathcal{H}}(\boldsymbol{m}+\boldsymbol{e})^T=\boldsymbol{\mathcal{H}}\boldsymbol{e}^T$. Le but est donc de trouver un vecteur $\boldsymbol{e}$ de poids, au plus, w (lié à la capacité du code) tel que $\boldsymbol{\mathcal{H}}\boldsymbol{e}^T=\boldsymbol{s}^T$.

Ces méthodes de décodage sont facilement adaptables à la recherche de mots de petit poids. En effet, il suffit de considérer le syndrome nul : $\boldsymbol{s}=\boldsymbol{0}$ pour que ces méthodes recherche un vecteur $\boldsymbol{e}$, de poids au plus w, tel que $\boldsymbol{\mathcal{H}}\boldsymbol{e}^T=\boldsymbol{0}^T$. Le vecteur $\boldsymbol{e}$ retrouvé est, par

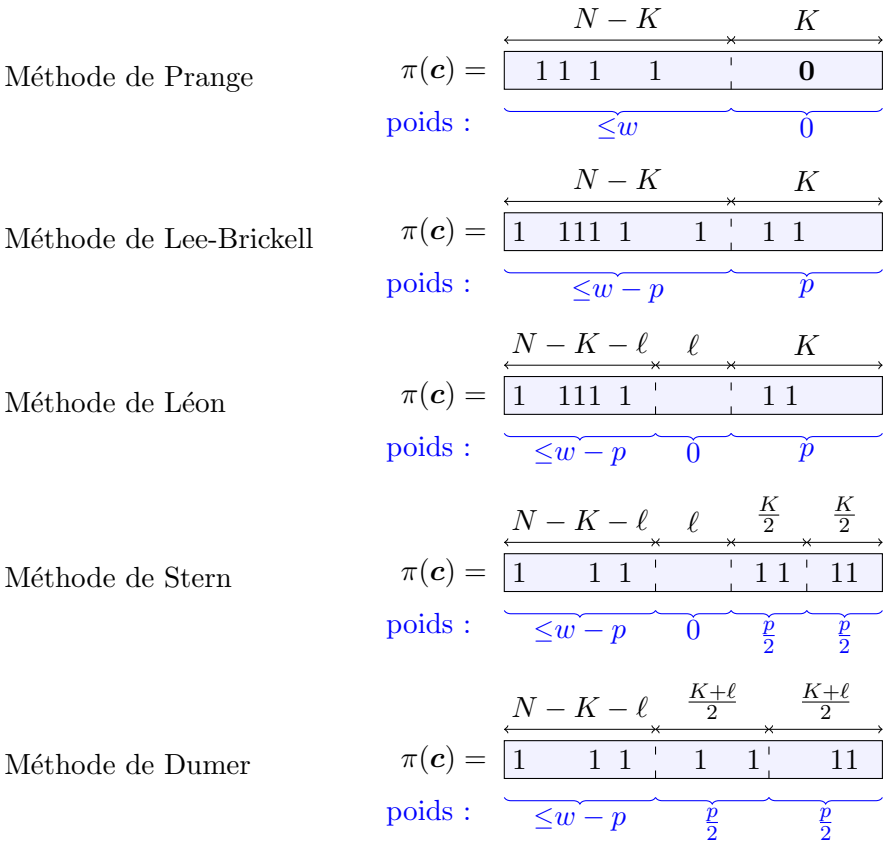


FIGURE 6.1 – Répartition de poids de $\pi(\boldsymbol{c})$ pour les différentes méthodes de type ISD

définition, un mot appartenant au code $\mathcal{C}$ et de petit poids w.

C'est de ce point de vue que sont présentées les méthodes de types ISD. Les vecteurs recherchés correspondent donc à des mots de code de petits poids, et non des vecteurs d'erreurs, c'est pour cette raison qu'ils sont notés $\boldsymbol{c}$. L'équation à résoudre est donc $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$ avec $\boldsymbol{c}$ de poids, au plus, w.

Les différentes méthodes, présentées ci-dessous, essayent de profiter du fait que les éléments non-nuls de $\boldsymbol{c}$ peuvent être localisés, en majorité, dans une fenêtre de taille inférieure ou égale à $N-K$. Pour cela, ces méthodes itératives choisissent aléatoirement une permutation π et recherchent s'il existe un mot de code $\boldsymbol{c}$ tel que la répartition de poids de $\pi(\boldsymbol{c})$ correspondent aux schémas reportés dans la figure 6.1. Notons que les deux dernières méthodes, celles de Stern et Dumer, utilisent en plus le fait qu'il est possible de rechercher plus efficacement qu'en $\mathcal{O}(\binom{K'}{p})$ tous les mots de poids p sur un ensemble de K' positions.

Méthode de Prange

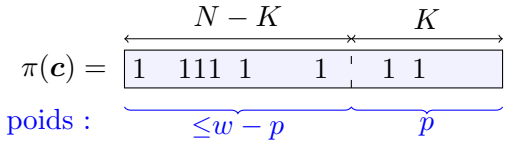
Pour résoudre le problème diverses méthodes existent, l'idée de départ date de 1962 et est due à Prange [111]. Dans cette méthode, un ensemble de K positions de $\boldsymbol{c}$ sont fixées à 0, ses $N-K$ autres positions sont ensuite recherchées telles que $\boldsymbol{c}$ soit de poids au plus w et satisfasse l'équation $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$. D'où, en notant π la permutation permettant de placer ces

K positions sur la droite, la répartition de poids de $\pi(\boldsymbol{c})$ donnée dans la table 6.1.

Cependant, la suite de cette méthode, prévue pour le décodage, ne s'applique pas directement à la recherche de mots de petit poids. En effet, le seul mot de code qui peut être retrouvé est le mot nul.

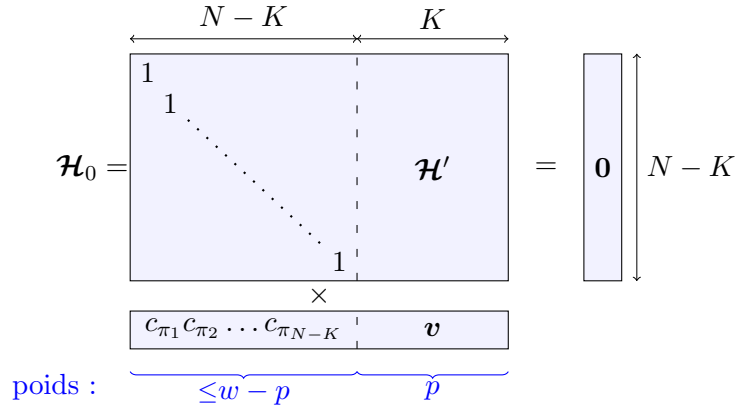
Méthode de Lee-Brickell

En 1988, Lee et Brickell [83] proposent de modifier légèrement la méthode de Prange. Avec cette modification la répartition de poids de $\pi(\boldsymbol{c})$ est moins contraignante. En effet, elle peut être représentée par :



De plus, la méthode obtenue permet, cette fois, de résoudre le problème de la recherche de mots de code de petits poids.

Plus précisément, K indices sont choisis aléatoirement, il est ensuite recherché les vecteurs $\boldsymbol{c}$, de poids p sur ces K indices, vérifiant $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$. Soit π la permutation permettant de placer ces K indices sur la droite de $\boldsymbol{c}$ et, afin d'alléger les notations, $\boldsymbol{c}_\pi = \pi(\boldsymbol{c})$. La matrice obtenue en appliquant la permutation π sur les colonnes de $\boldsymbol{\mathcal{H}}$ est notée $\boldsymbol{\mathcal{H}}_\pi$. Un pivot de Gauss est appliqué sur la matrice $\boldsymbol{\mathcal{H}}_\pi$ afin d'obtenir la matrice $\boldsymbol{\mathcal{H}}_0$. Finalement, en notant $\boldsymbol{v}$ le vecteur de longueur K défini par $\boldsymbol{v} = (c_{\pi_{N-K+1}}, \ldots, c_{\pi_N})$ la situation peut être représentée de la façon suivante :



Si $\boldsymbol{\mathcal{H}}_0\boldsymbol{c}_\pi^T = \mathbf{0}^T$ alors l'égalité $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$ est vérifiée. De plus, afin que l'équation $\boldsymbol{\mathcal{H}}_0\boldsymbol{c}_\pi^T = \mathbf{0}^T$ soit vérifiée, la structure de $\boldsymbol{\mathcal{H}}_0$ implique que $(c_{\pi_1}, \ldots, c_{\pi_{N-K}})^T = \boldsymbol{\mathcal{H}}'\boldsymbol{v}^T$. D'où si $\boldsymbol{\mathcal{H}}'\boldsymbol{v}^T$ est de poids au plus $w - p$, alors le vecteur $\boldsymbol{c} = \pi^{-1}(\boldsymbol{\mathcal{H}}'\boldsymbol{v}^T|\boldsymbol{v})$ est un mot de code de poids au plus w (où $\mathbf{a}|\mathbf{b}$ représente la concaténation des vecteurs $\mathbf{a}$ et $\mathbf{b}$).

La méthode de Lee-Brickell pour retrouver un mot de code $\boldsymbol{c}$ de poids au plus w s'en déduit :

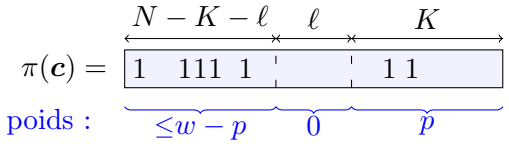
Tant que nécessaire itérer :

1. Choisir aléatoirement une permutation π
2. Déterminer $\boldsymbol{\mathcal{H}}_\pi$ obtenue en appliquant π sur les colonnes de $\boldsymbol{\mathcal{H}}$
3. Calculer $\boldsymbol{\mathcal{H}}_0$ en appliquant un pivot de Gauss à la matrice $\boldsymbol{\mathcal{H}}_\pi$
4. Pour tous les vecteurs $\boldsymbol{v}$ de longueur K et de poids p
   Si $\boldsymbol{\mathcal{H}}'\boldsymbol{v}^T$ est de poids $\leq w - p$ alors
   Retourner $\pi^{-1}(\boldsymbol{\mathcal{H}}'\boldsymbol{v}^T|\boldsymbol{v})$

Pour minimiser le coût de la recherche, la valeur optimale pour p est de l'ordre de 2 pour des dimensions de code "courantes".

Méthode de Léon

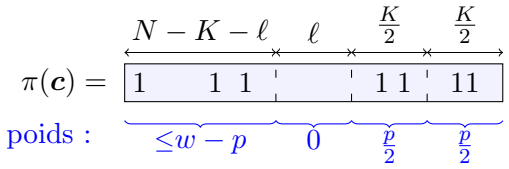
Toujours en 1988, Léon [84] propose une amélioration de la méthode de Lee-Brickell. Dans cette méthode la contrainte sur la répartition de poids de $\pi(\boldsymbol{c})$ est plus importante qu'avec la méthode de Lee-Brickell. En effet, afin de retrouver un mot de code $\boldsymbol{c}$ lors d'une itération il faut que $\pi(\boldsymbol{c})$ possède la répartition de poids :



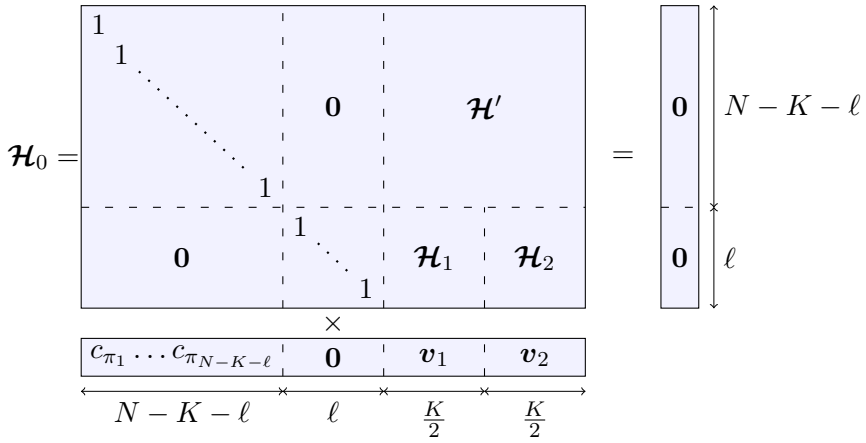
Cette répartition diminue la probabilité de trouver un mot de code lors d'une itération mais permet de réduire le coût de chaque itération, la valeur de $\boldsymbol{\mathcal{H}}'\boldsymbol{v}^T$ n'étant pas entièrement calculée (si elle ne possède pas les 0 souhaités, le vecteur $\boldsymbol{v}$ est rejeté et l'algorithme passe au suivant).

Méthode de Stern

En utilisant le paradoxe des anniversaires Stern [122] améliore encore la méthode de Léon. Avec cette méthode le vecteur $\pi(\boldsymbol{c})$ doit posséder la répartition de poids :



Cette répartition de poids est plus contraignante que celle de la méthode de Léon mais le gain de temps à chaque itération la compense largement. La matrice $\boldsymbol{\mathcal{H}}_0$, est partagée en sous-matrices comme sur le schéma suivant :

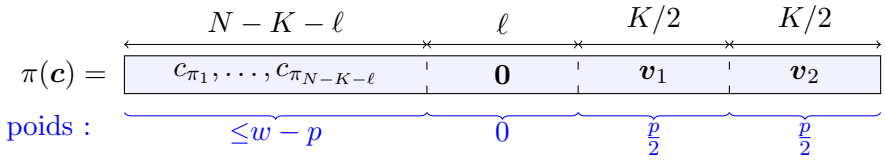


Afin de trouver le vecteur $\boldsymbol{c}_\pi$ possédant la répartition de poids précédente, deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$ sont crées telles que : $\mathcal{L}_i$ contient toutes les combinaisons linéaires de $p/2$ colonnes de $\boldsymbol{\mathcal{H}}_i$. Ainsi :

$$\mathcal{L}_1 = \{(\boldsymbol{v}_1, \boldsymbol{\mathcal{H}}_1 \boldsymbol{v}_1^T) | \boldsymbol{v}_1 \text{ est de taille } K/2 \text{ et de poids p}\}$$

$$\mathcal{L}_2 = \{(\boldsymbol{v}_2, \boldsymbol{\mathcal{H}}_2 \boldsymbol{v}_2^T) | \boldsymbol{v}_2 \text{ est de taille } K/2 \text{ et de poids p}\}$$

Soient deux couples $(\boldsymbol{v}_1, \boldsymbol{\mathcal{H}}_1 \boldsymbol{v}_1^T)$, et $(\boldsymbol{v}_2, \boldsymbol{\mathcal{H}}_2 \boldsymbol{v}_2^T)$ appartenant respectivement à $\mathcal{L}_1$ et $\mathcal{L}_2$ tels que $\boldsymbol{\mathcal{H}}_1 \boldsymbol{v}_1^T = \boldsymbol{\mathcal{H}}_2 \boldsymbol{v}_2^T$. Pour que le vecteur $\boldsymbol{c}_\pi = \pi(\boldsymbol{c})$ défini par :



satisfasse les conditions souhaitées, il suffit que $(c_{\pi_1}, \ldots, c_{\pi_{N-K-\ell}})^T = \boldsymbol{\mathcal{H}}'(\boldsymbol{v}_1 | \boldsymbol{v}_2)^T$ et soit de poids inférieur ou égal à $w-p$. Ainsi, l'étape 4. de l'algorithme devient :

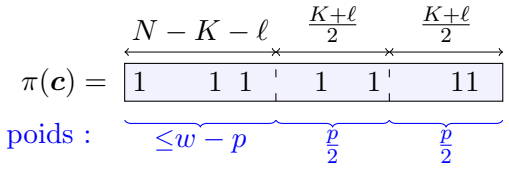
4. Créer les listes $\mathcal{L}_1$ et $\mathcal{L}_2$
   Pour tous les couples $(\boldsymbol{v}_1, \boldsymbol{v}_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ tels que $\boldsymbol{\mathcal{H}}_1 \boldsymbol{v}_1^T = \boldsymbol{\mathcal{H}}_2 \boldsymbol{v}_2^T$
   Si $\boldsymbol{v}^T = \boldsymbol{\mathcal{H}}'(\boldsymbol{v}_1 | \boldsymbol{v}_2)^T$ est de poids, au plus, $w-p$ alors
   Retourner $\pi^{-1}(\boldsymbol{v} | \boldsymbol{0} | \boldsymbol{v}_1 | \boldsymbol{v}_2)$

Cette façon de procéder est plus efficace que la méthode de Léon. En effet, dans la méthode de Léon, comme dans celle de Lee et Brickell, la complexité de la méthode est dominée par le test de tous les vecteurs de taille K et de poids p. La complexité de ces méthodes est donc en $\mathcal{O}(\binom{K}{p})$, alors que pour la méthode proposée par Stern la complexité est en $\mathcal{O}(\binom{K/2}{p/2})$.

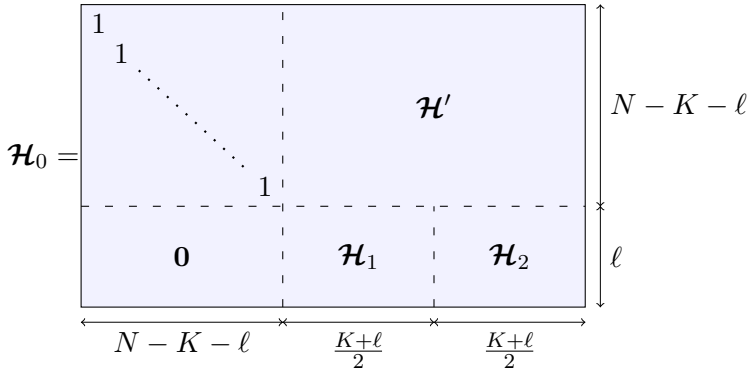
Grâce à un petit changement sur cette méthode, Dumer [39] en propose une nouvelle encore plus efficace en 1991.

Méthode de Dumer

Dans la méthode proposée par Dumer [39] la répartition de poids de $\pi(\boldsymbol{c})$ est moins stricte que celle de la méthode de Stern. Cette répartition est de la forme :



Pour cela, seul le découpage de la matrice $\boldsymbol{\mathcal{H}}_0$ est modifié, les ℓ colonnes se trouvant au milieu sont rattachées à la partie de droite. La situation peut alors être représentée par :

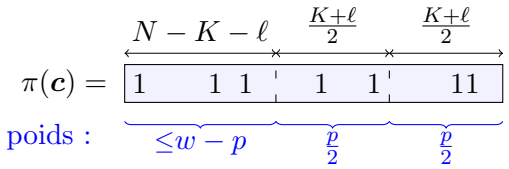


La suite de l'algorithme est exactement la même que pour la méthode de Stern.

Le choix des paramètres p et ℓ est important afin de limiter le coût de la recherche des mots de code de petits poids. Pour déterminer les valeurs à utiliser, le coût de la recherche d'un mot de code donné est calculé, il est noté WF (Work Factor).

$$WF = \frac{\text{Coût d'une itération}}{\text{Probabilité de trouver un mot de code lors d'une itération}} = \frac{C}{P}$$

La probabilité P de trouver un mot de code $\boldsymbol{c}$ lors d'une itération est égale à la probabilité que $\pi(\boldsymbol{c})$ ait la répartition de poids :



La permutation π est aléatoire, P est donc égale à la probabilité que $\boldsymbol{c}$ ait cette répartition de poids. Autrement dit, P est égale au nombre de vecteurs de poids w respectant la répartition de poids souhaitée divisée par le nombre de vecteurs de poids w, d'où :

$$P = \frac{\binom{N-K-\ell}{w-p}\binom{(K+\ell)/2}{p/2}^2}{\binom{N}{w}}$$

Pour le calcul du coût C d'une itération, l'opération élémentaire considérée est la somme de deux colonnes ou le calcul du poids d'une colonne. C dépend des coûts du pivot de Gauss sur les $N-K-\ell$ premières lignes de $\boldsymbol{\mathcal{H}}_\pi$, de la fabrication des deux listes, de la recherche de l'intersection ainsi que du coût de la vérification du poids de $\boldsymbol{\mathcal{H}}'(\boldsymbol{v}_1|\boldsymbol{v}_2)^T$. Plus précisément ces différents coûts sont :

— Le coût du pivot de Gauss est $(N-K-\ell)N$.
— Chaque liste contient $\binom{(K+\ell)/2}{p/2}$ éléments, le coût de la construction des deux listes est environ de $2\binom{(K+\ell)/2}{p/2}$.
— L'intersection des deux listes est de taille $\frac{\binom{(K+\ell)/2}{p/2}^2}{2^\ell}$.
— Tester un élément de l'intersection consiste à effectuer une somme de colonnes et à déterminer son poids ; le coût de cette vérification est essentiellement constant et estimé à 2.

D'où le coût d'une itération :

$$C = (N-K-\ell)N + 2\binom{(K+\ell)/2}{p/2} + 2\frac{\binom{(K+\ell)/2}{p/2}^2}{2^\ell}$$

Afin de simplifier le calcul des paramètres optimaux, le coût du pivot de Gauss n'est pas pris en compte. De plus, il est négligeable devant les coûts concernant les listes dès que p est supérieur ou égal à 8.

Une valeur approchée de WF en découle :

$$\begin{aligned} WF &\simeq \frac{2\binom{(K+\ell)/2}{p/2} + 2\frac{\binom{(K+\ell)/2}{p/2}^2}{2^\ell}}{\frac{\binom{R-\ell}{w-p}\binom{(K+\ell)/2}{p/2}^2}{\binom{N}{w}}} \\ &= \frac{\binom{N}{w}}{\binom{N-K-\ell}{w-p}}\left(\frac{2}{\binom{(K+\ell)/2}{p/2}} + \frac{2}{2^\ell}\right) \end{aligned}$$

Pour minimiser WF, il faut minimiser $\frac{2}{\binom{(K+\ell)/2}{p/2}} + \frac{2}{2^\ell}$. À un petit facteur près ce terme est minimal lorsque $\frac{2}{\binom{(K+\ell)/2}{p/2}} = \frac{2}{2^l}$, et dans ce cas :

$$WF = \frac{4\binom{N}{w}}{\binom{N-K-\ell}{w-p}2^\ell}$$

Il faut donc choisir p et ℓ tels que $\frac{\binom{N}{w}}{\binom{N-K-\ell}{w-p}2^\ell}$ est minimal et tels que $2^\ell \simeq \binom{(K+\ell)/2}{p/2}$.

Remarque. *Il existe d'autre méthodes de décodages de type ISD, telles que [7] et [105] mais elles sont nettement plus compliquées à mettre en œuvre.*

6.2 Recherche des équations de parité de petit poids

La méthode de Valembois, présentée précédemment, recherche des mots de petits poids appartenant au code généré par la matrice :

$$\mathcal{X} = \begin{pmatrix} \tilde{\boldsymbol{m}}^{1^T} & \tilde{\boldsymbol{m}}^{2^T} & \cdots & \tilde{\boldsymbol{m}}^{M^T} \end{pmatrix}$$

De ces mots de code sont déduites des équations de parité, leurs poids est vérifié ultérieurement, il n'est pas forcément égal à t.

Dans la méthode de Cluzeau-Finiasz, les équations de parité retrouvées sont nécessairement de poids t et le poids du mot de code est utilisé pour valider, ou non, une équation de parité.

Afin de généraliser ces deux méthodes, l'idée que l'on va utiliser est de rechercher directement des équations de parité $\boldsymbol{h}$ de poids t, telles que le mot de code $\boldsymbol{c} = \boldsymbol{h}\mathcal{X}$ soit de petit poids. Pour cela, il suffit de rechercher des mots de petit poids dans le code généré par la matrice obtenue en concaténant une matrice identité et la matrice $\mathcal{X}$. En effet :

Soient $\mathcal{C}_1$ un code (N, K) et un ensemble de M mots de ce code bruités par un canal binaire symétrique de probabilité τ : $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ où $\tilde{\boldsymbol{m}}^i = (\tilde{m}^i_1, \tilde{m}^i_2, \ldots, \tilde{m}^i_N)$.

Soit $\mathcal{C}_2$ le code $(N + M, N)$ ayant la matrice $\boldsymbol{\mathcal{G}}_{\mathcal{C}_2}$ pour matrice génératrice :

$$\boldsymbol{\mathcal{G}}_{\mathcal{C}_2} = \left(\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & \tilde{\boldsymbol{m}}^{1T} & \tilde{\boldsymbol{m}}^{2T} & \cdots \quad \tilde{\boldsymbol{m}}^{MT} \\ & & 1 & & & \end{array}\right)$$

Proposition 6.1. *Soient $\boldsymbol{h} = (h_1, \ldots, h_N)$ un vecteur de poids t*

— *Si $\boldsymbol{h}$ est une équation de parité du code $\mathcal{C}_1$, c'est-à-dire un mot appartenant au code dual de $\mathcal{C}_1$, alors, pour tout i :*

$$\boldsymbol{P}(\boldsymbol{h} \times (\tilde{\boldsymbol{m}}^i)^T = 1) = \frac{1 - (1 - 2\tau)^t}{2}$$

— *Sinon (c'est-à-dire $\boldsymbol{h}$ n'est pas une équation de parité du code $\mathcal{C}_1$), alors pour tout i :*

$$\boldsymbol{P}(\boldsymbol{h} \times (\tilde{\boldsymbol{m}}^i)^T = 1) = \frac{1}{2}$$

De cette proposition, il se déduit que si $\boldsymbol{c} = (c_1, \ldots, c_{N+M})$ est un mot du code $\mathcal{C}_2$ défini par $\boldsymbol{c} = \boldsymbol{h}\boldsymbol{\mathcal{G}}_{\mathcal{C}_2}$, avec $\boldsymbol{h}$ un vecteur de petit poids t, alors :

— Si $\boldsymbol{h}$ est une équation de parité du code $\mathcal{C}_1$, le vecteur $\boldsymbol{c}$ est de poids faible et possède la répartition de poids :

$$\boldsymbol{c} = (\underbrace{c_1, \ldots, c_N}_{=t}, \underbrace{c_{N+1}, \ldots, c_{N+M}}_{\simeq \frac{1-(1-2\tau)^t}{2} M})$$

— Sinon, $\boldsymbol{c}$ est de poids élevé dont la répartition de poids est :

$$\boldsymbol{c} = (\underbrace{c_1, \ldots, c_N}_{=t}, \underbrace{c_{N+1}, \ldots, c_{N+M}}_{\simeq \frac{1}{2} M})$$

Afin de trouver des équations de parité de poids t vérifiées par les mots du code $\mathcal{C}_1$, une méthode consiste donc à rechercher des mots $\boldsymbol{c}$ appartenant au code $\mathcal{C}_2$ et ayant pour répartition de poids :

$$\boldsymbol{c} = (\underbrace{c_1, \ldots, c_N}_{=t}, \underbrace{c_{N+1}, \ldots, c_{N+M}}_{\simeq \frac{1-(1-2\tau)^t}{2} M})$$

Le problème de la recherche des équations de parité de petit poids dans le code $\mathcal{C}_1$ est ainsi ramené au problème de recherche de mots de petits poids dans le code $\mathcal{C}_2$ (avec une

contrainte sur leur répartition de poids).

La matrice génératrice du code $\mathcal{C}_2$ étant systématique, la matrice $\boldsymbol{\mathcal{H}}_{\mathcal{C}_2}$, définie ci-dessous, est donc une matrice de parité de ce code.

$$\boldsymbol{\mathcal{H}}_{\mathcal{C}_2} = \left(\begin{array}{c|ccc} \tilde{\boldsymbol{m}}^1 & 1 & & \\ \tilde{\boldsymbol{m}}^2 & & 1 & \\ \vdots & & & \ddots \\ \tilde{\boldsymbol{m}}^M & & & 1 \end{array}\right)$$

La recherche des équations de parité de petit poids peut donc être effectuée grâce aux méthodes présentées dans la section 6.1.4. Des mots de poids au plus w, avec w légèrement supérieur à $t + \frac{1-(1-2\tau)^t}{2}M$, vont être recherchés à partir de la matrice de parité $\boldsymbol{\mathcal{H}}_{\mathcal{C}_2}$.

6.3 Utilisation de l'information concernant le poids de $\boldsymbol{c}$

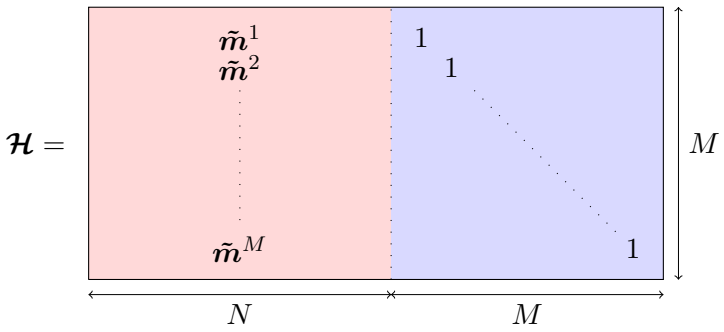
En notant $\boldsymbol{\mathcal{H}}$ la matrice $\boldsymbol{\mathcal{H}}_{\mathcal{C}_2}$ de la section précédente, la méthode de Dumer, comme les autres méthodes, recherche des vecteurs $\boldsymbol{c}$ de poids au plus w tels que $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \boldsymbol{0}^T$. Avec ces méthodes, le poids de $\boldsymbol{c}$ est contrôlé, il doit être inférieur à w, mais aucune contrainte sur sa répartition de poids n'est prise en compte. En effet, même si la répartition de poids du vecteur $\boldsymbol{c}_\pi$ est connue, il est impossible d'en déduire celle de $\boldsymbol{c}$ sans déterminer $\pi^{-1}(\boldsymbol{c}_\pi)$, la permutation π étant aléatoire. Or la répartition de poids de $\boldsymbol{c}$ est connue à priori, il est donc intéressant de l'utiliser afin d'accélérer la recherche.

Une modification à apporter à l'algorithme de Dumer afin de trouver plus rapidement des mots $\boldsymbol{c}$ du code $\mathcal{C}_2$ ayant la répartition de poids

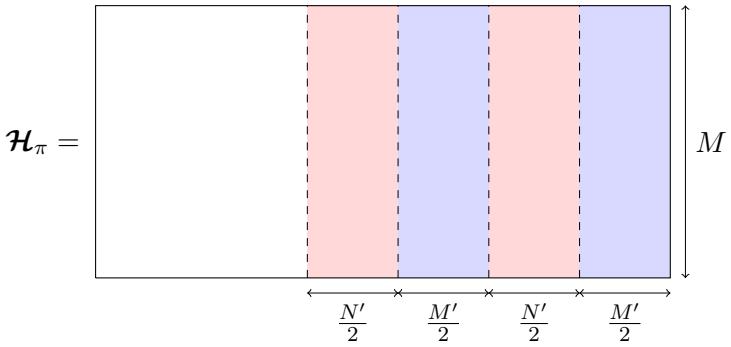
$$\boldsymbol{c} = (\underbrace{c_1, \ldots, c_N}_{=t}, \underbrace{c_{N+1}, \ldots, c_{N+M}}_{\simeq \frac{1-(1-2\tau)^t}{2}M})$$

porte donc sur la permutation des colonnes à effectuer au début de chaque itération. Cela impacte également la construction des deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$.

La matrice de parité $\boldsymbol{\mathcal{H}}$ est de taille $M \times (M+N)$ et est composée de deux parties distinctes, la partie de gauche contenant les mots bruités et celle de droite qui est une matrice identité.

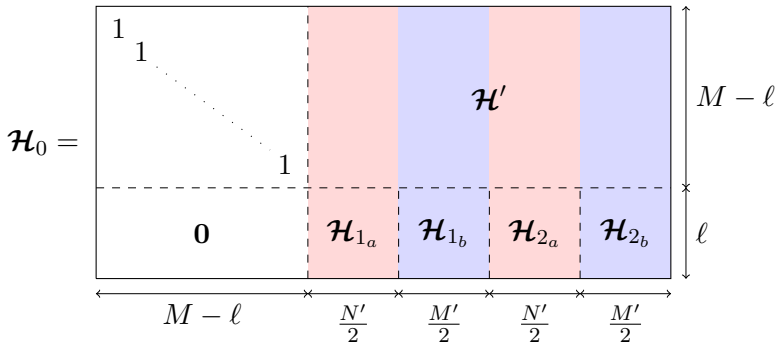


Il est possible de choisir une permutation π telle qu'en l'appliquant aux colonnes de $\boldsymbol{\mathcal{H}}$ la matrice obtenue $\boldsymbol{\mathcal{H}}_\pi$ soit de la forme



où des colonnes rouges sont des colonnes venant de la partie de gauche de $\boldsymbol{\mathcal{H}}$ et les colonnes en bleues de la partie identité de $\boldsymbol{\mathcal{H}}$. Les $N + M - N' - M'$ premières colonnes de $\boldsymbol{\mathcal{H}}_\pi$ sont un mélange de colonnes venant des deux parties de $\boldsymbol{\mathcal{H}}$.

Comme dans la version initiale de l'algorithme de Dumer, afin d'obtenir la matrice $\boldsymbol{\mathcal{H}}_0$, un pivot de Gauss est effectué sur les $M - \ell$ premières lignes de $\boldsymbol{\mathcal{H}}_\pi$. La matrice $\boldsymbol{\mathcal{H}}_0$ peut alors être représentée par :

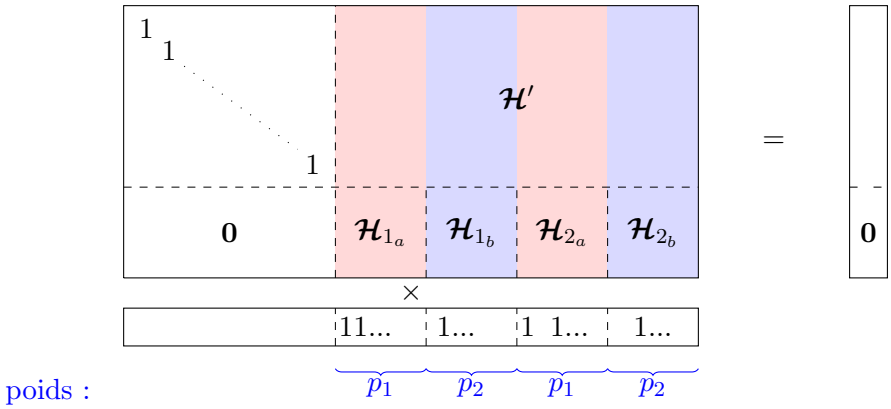


Dans l'algorithme présenté précédemment, deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$ sont construites en effectuant toutes les combinaisons linéaires de $p/2$ colonnes respectivement des matrices $\boldsymbol{\mathcal{H}}_1$ et $\boldsymbol{\mathcal{H}}_2$. Dans cette version modifiée de l'algorithme, la liste $\mathcal{L}_i$ est composée de toutes les combinaisons linéaires de p_1 colonnes de $\boldsymbol{\mathcal{H}}_{i_a}$ et p_2 colonnes de $\boldsymbol{\mathcal{H}}_{i_b}$, pour $i = 1, 2$.

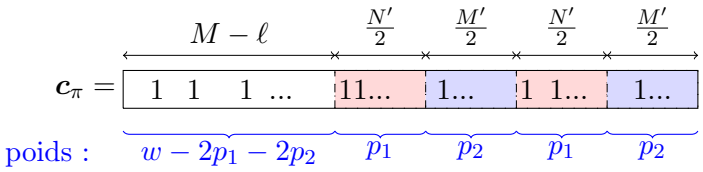
$$\mathcal{L}_i = \{(\boldsymbol{u}|\boldsymbol{v}, \boldsymbol{\mathcal{H}}_{i_a}\boldsymbol{u}^T + \boldsymbol{\mathcal{H}}_{i_b}\boldsymbol{v}^T) | \boldsymbol{u} \text{ est de poids } p_1 \text{ et } \boldsymbol{v} \text{ est de poids } p_2\}$$

(avec $\boldsymbol{u}|\boldsymbol{v}$ la concaténation des vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$).

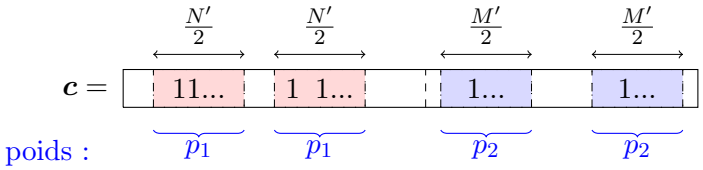
L'intersection des deux listes est ensuite déterminée afin de trouver un ensemble de mots de code potentiellement de petits poids. Ces mots sont tels que :



Le poids total de ces mots est ensuite vérifié, de la même façon qu'auparavant. La répartition de poids des mots $\boldsymbol{c}_\pi$ tels que $\boldsymbol{\mathcal{H}}_0\boldsymbol{c}_\pi^T = \mathbf{0}^T$ qui sont ainsi trouvés est donc la suivante :



En appliquant π^{-1} à $\boldsymbol{c}_\pi$ un vecteur $\boldsymbol{c}$ tel que $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$ est déduit. Par construction il possède la répartition de poids :

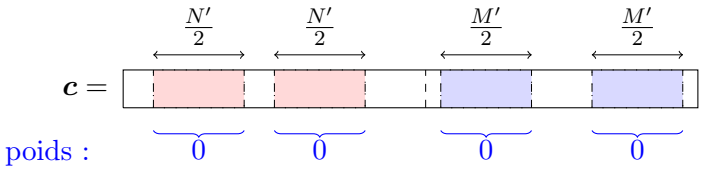


En procédant de cette façon et avec de bons paramètres M', N', p_1 et p_2, la probabilité que $\boldsymbol{c}$ respecte la répartition de poids

$$\boldsymbol{c} = (\underbrace{c_1, \ldots, c_N}_{=t}, \underbrace{c_{N+1}, \ldots, c_{N+M}}_{\simeq \frac{1-(1-2\tau)^t}{2} M})$$

est maximisée tout comme la probabilité de trouver à chaque itération un tel mot de code.

Remarquons qu'en prenant en compte les équations qui se déduisent directement de l'étape du pivot de Gauss, l'algorithme obtenu généralise la méthode de Gauss Randomisé mais accélère également la recherche des équations de petit poids, en particulier lorsque les données sont très faiblement bruitées. En effet, cela permet de retrouver des équations dont la répartition de poids est :



alors que sans cette modification, l'algorithme ne retient pas ces équations et recherche une permutation π telle qu'il n'y a pas d'équation possédant la répartition de poids donnée ci-dessus. Cette adaptation n'a pas été effectuée dans les programmes de tests, les temps présentés dans les prochains chapitres ne la prennent donc pas en compte.

Si un vecteur $\boldsymbol{c} = (c_1, \ldots, c_{N+M})$ de poids au plus w et tel que $\boldsymbol{\mathcal{H}}\boldsymbol{c}^T = \mathbf{0}^T$ est retrouvé alors, avec une grande probabilité, le vecteur $(c_1, \ldots, c_N) = \boldsymbol{h} = (h_1, \ldots, h_N)$ définit une des équations de parité recherchées.

6.4 Choix des paramètres

Afin de maximiser la probabilité de retrouver des mots de petit poids, la permutation π à appliquer aux colonnes de la matrice $\boldsymbol{\mathcal{H}}$ au début de chaque itération n'est pas choisie aléatoirement. Cette permutation dépend de N' et M', ces paramètres dépendant eux mêmes de N et M, mais également de p_1 et p_2 ainsi que du niveau de bruit τ. Cette section est consacrée au choix optimal de ces paramètres afin de minimiser le coût de la recherche des mots de petit poids.

6.4.1 Coût de la méthode de Dumer modifiée

Avec ce contrôle sur la permutation, le coût pour retrouver un mot de code donné de petit poids est réduit. Comme pour la version classique, le coût de la méthode est donné par :

$$WF = \frac{\text{Coût d'une itération}}{\text{Probabilité que le mot cherché respecte la répartition de poids voulue}}$$

De même que précédemment, l'opération élémentaire considérée est la somme de deux colonnes ou le calcul du poids d'une colonne. Le coût du pivot de Gauss est considéré comme constant et n'est pas pris en compte dans le calcul de WF. Le coût d'une itération correspond donc au calcul des deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$, de leur intersection et du test des mots potentiellement de petit poids.

Il peut être noté que le coût du pivot de Gauss n'est pas négligeable dans notre cas. En effet, les valeurs p_1 et p_2 sont petites (au maximum égales à 2), les coûts liés aux listes ne dominent alors pas le coût du pivot. Cependant, le supposer constant permet de simplifier les calculs et permet de comprendre plus facilement le comportement des paramètres optimaux en fonction de N, M et τ.

Proposition 6.2. *Le coût de l'algorithme de Dumer modifié est :*

$$WF = \frac{2\binom{N'/2}{p_1}\binom{M'/2}{p_2} + 2\frac{\left(\binom{N'/2}{p_1}\binom{M'/2}{p_2}\right)^2}{2^{N'+M'-N}}}{\frac{\binom{N'/2}{p_1}^2\binom{N-N'}{t-2p_1}}{\binom{N}{t}}\left(\binom{M'/2}{p_2}\left(\frac{1-(1-2\tau)^t}{2}\right)^{p_2}\left(\frac{1+(1-2\tau)^t}{2}\right)^{M'/2-p_2}\right)^2}$$

Démonstration. Les deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$ sont de taille $\binom{N'/2}{p_1}\binom{M'/2}{p_2}$. Le coût de la fabrication de ces listes et de leur intersection est donc de $2\binom{N'/2}{p_1}\binom{M'/2}{p_2}$.

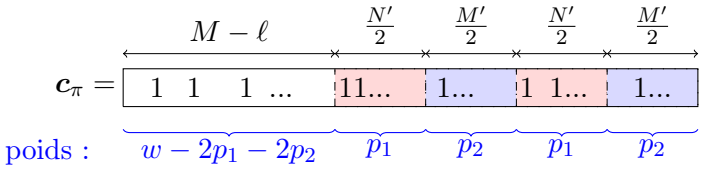
Quant-à l'intersection de $\mathcal{L}_1$ et $\mathcal{L}_2$ elle est de taille :

$$\frac{\left(\binom{N'/2}{p_1}\binom{M'/2}{p_2}\right)^2}{2^{N'+M'-N}}$$

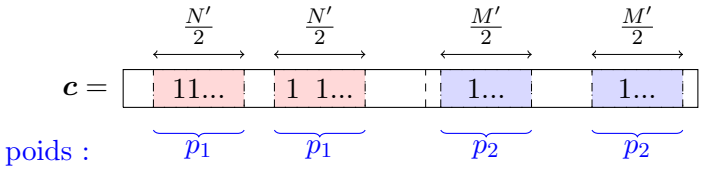
En effet, par construction, ℓ est tel que $M-\ell+N'+M'=M+N$ d'où $\ell=N'+M'-N$. Le test d'un élément de l'intersection à un coût de 2 opérations élémentaires. Le coût C d'une itération en découle :

$$C = 2\binom{N'/2}{p_1}\binom{M'/2}{p_2} + 2\frac{\left(\binom{N'/2}{p_1}\binom{M'/2}{p_2}\right)^2}{2^{N'+M'-N}}$$

La probabilité P que $\boldsymbol{c}_\pi$ ait la répartition de poids :



est égale à la probabilité que $\boldsymbol{c}$ suive la répartition de poids suivante :



Afin de déterminer P son calcul est partagé en deux termes P_{gauche} et P_{droite}. Le premier correspond à la probabilité que $\boldsymbol{c}$ respecte la répartition de poids sur la partie de gauche, c'est-à-dire sur les N premiers indices, P_{droite} concerne donc la partie droite (i.e. sur les M derniers indices). P vérifie donc $P = P_{gauche} \times P_{droite}$.

Par hypothèse, $\boldsymbol{c}$ est de poids exactement t sur les N premiers indices. La probabilité P_{gauche} que $\boldsymbol{c}$ respecte la répartition de poids sur ces N premiers indices (c'est-à-dire de poids exactement p_1 pour chacun des deux ensembles de $N'/2$ positions choisies) est donc égale à :

$$P_{gauche} = \frac{\binom{N'/2}{p_1}^2\binom{N-N'}{t-2p_1}}{\binom{N}{t}}$$

Sur la partie de droite, le poids de $\boldsymbol{c}$ n'est pas connu précisément, cependant la probabilité qu'un indice donné soit non nul est connue ; elle est égale à $\frac{1-(1-2\tau)^t}{2}$. D'où :

$$P_{droite} = \left(\binom{M'/2}{p_2}\left(\frac{1-(1-2\tau)^t}{2}\right)^{p_2}\left(\frac{1+(1-2\tau)^t}{2}\right)^{M'/2-p_2}\right)^2$$

Du coût C et des probabilités P_{gauche} et P_{droite} le coût de l'algorithme se déduit :

$$WF = \frac{2\binom{N'/2}{p_1}\binom{M'/2}{p_2} + 2\frac{\left(\binom{N'/2}{p_1}\binom{M'/2}{p_2}\right)^2}{2^{N'+M'-N}}}{\frac{\binom{N'/2}{p_1}^2\binom{N-N'}{t-2p_1}}{\binom{N}{t}}\left(\binom{M'/2}{p_2}\left(\frac{1-(1-2\tau)^t}{2}\right)^{p_2}\left(\frac{1+(1-2\tau)^t}{2}\right)^{M'/2-p_2}\right)^2}$$

□

Il peut être remarqué qu'en considérant le coût du pivot de Gauss comme constant le nombre M de mots de code bruités n'influe pas directement le coût de cette méthode. Par contre, M impacte directement le coût du pivot ainsi que la valeur de w, le choix de M est donc important pour minimiser le coût de la recherche des mots de petits poids.
Une autre variable qui n'intervient pas dans cette formule est K la dimension du code recherché. Or si $N - N' > K$ et que les mots observés ne sont pas ou peu bruités, il est impossible d'obtenir l'identité sur les $M - \ell$ premières colonnes de $\boldsymbol{\mathcal{H}}_0$. En pratique, de cette impossibilité il se déduit tout de même un ensemble d'équations de parité de petit poids lors du pivot de Gauss, de la même façon que dans la méthode de Gauss Randomisé.

6.4.2 Calcul des paramètres optimaux

En continuant de négliger le coût du pivot de Gauss et en faisant varier les différents paramètres, le coût de l'algorithme semble généralement être minimal pour $p_1 = 1$ et $p_2 = 0$ (d'après des simulations faites sur Maple). Le théorème suivant détermine les paramètres optimaux N' et M' lorsque $p_1 = 1$ et $p_2 = 0$ en fonction de N, τ et t le poids des mots de code recherchés.

Théorème 6.3. *Le coût de la recherche d'un mot de code de poids t lorsque $p_1 = 1$ et $p_2 = 0$ est minimal pour :*

$$\begin{cases} M' &= \frac{\log(N')}{\log(2)} - N' + N - 1 \\ N' &= min(N, N'_1) \end{cases}$$

Avec

$$N'_1 = \frac{\log(1 + (1-2\tau)^t) + (t-2) + N\log\left(\frac{1+(1-2\tau)^t}{2}\right) - \sqrt{\Delta}}{2\log\left(\frac{1+(1-2\tau)^t}{2}\right)}$$

et

$$\Delta = \left(2 - t - \log(1 + (1-2\tau)^t) - N\log\left(\frac{1+(1-2\tau)^t}{2}\right)\right)^2 - 4N\log\left(\frac{1+(1-2\tau)^t}{2}\right)\log(1 + (1-2\tau)^t)$$

Corollaire 6.4. *Le coût de l'algorithme sous l'hypothèse de $p_1 = 1$ et $p_2 = 0$ est :*

$$WF \simeq \frac{1}{N'\binom{N-N'}{t-2}\left(\frac{1+(1-2\tau)^t}{2}\right)^{M'}} \times 8\binom{N}{t}$$

Dans les preuves suivantes, la notation $T = \frac{1+(1-2\tau)^t}{2}$ est utilisée, c'est une constante ne dépendant pas des paramètres de la méthode.

Démonstration du corollaire 6.4. Lorsque $p_1 = 1$ et $p_2 = 0$, le coût C d'une itération est de :

$$\begin{aligned} C &= 2\binom{N'/2}{1}\binom{M'/2}{0} + 2\frac{\left(\binom{N'/2}{1}\binom{M'/2}{0}\right)^2}{2^{N'+M'-N}} \\ &= N' + \frac{N'^2}{2}\frac{1}{2^{N'+M'-N}} \\ &= N'\left(1 + \frac{N'}{2^{N'+M'-N+1}}\right) \end{aligned}$$

Pour minimiser ce coût, il faut choisir N tel que $N' \simeq 2^{N'+M'-N+1}$, ce qui implique $C \simeq 2N'$.

La probabilité P que le mot recherché respecte la répartition de poids est :

$$\begin{aligned} P &= \frac{\binom{N'/2}{1}^2 \binom{N-N'}{t-2}}{\binom{N}{t}} \left(\binom{M'/2}{0} (1-T)^0 T^{M'/2-0} \right)^2 \\ &= \frac{N'^2 \binom{N-N'}{t-2}}{4\binom{N}{t}} T^{M'} \end{aligned}$$

Le coût de l'algorithme sous l'hypothèse de $p_1 = 1$ et $p_2 = 0$ est donc :

$$\begin{aligned} WF &\simeq \frac{C}{P} \\ &= \frac{2N'}{\binom{N-N'}{t-2} N'^2 T^{M'}} \times 4\binom{N}{t} \\ &= \frac{1}{\binom{N-N'}{t-2} N' T^{M'}} \times 8\binom{N}{t} \end{aligned}$$

□

Démonstration du théorème 6.3. L'objectif est de minimiser WF :

$$WF \simeq \frac{1}{N'\binom{N-N'}{t-2} T^{M'}} \times \underbrace{8\binom{N}{t}}_{\text{constante}}$$

il faut donc maximiser son dénominateur :

$$\begin{aligned} & N' \binom{N-N'}{t-2} T^{M'} \\ \simeq\ & N'(N-N')^{t-2} T^{M'} \end{aligned} \tag{6.1}$$

Le but est donc de maximiser $\log(N') + (t-2)\log(N-N') + M'\log(T)$. Afin de minimiser le coût de chaque itération, N' est choisi tel que $N' \simeq 2^{N'+M-N+1}$ (hypothèse faite dans la démonstration du corollaire 6.4). D'où $\log(N') = (N' + M' - N + 1)\log(2)$ et donc $M' = \frac{\log(N')}{\log(2)} - N' + N - 1$.

En remplaçant M' dans l'équation (6.1), la fonction en N' à maximiser est :

$$f(N') = \log(N') + (t-2)\log(N-N') + \left(\frac{\log(N')}{\log(2)} - N' + N - 1 \right) \log(T)$$

Les variations de f sont étudiées grâce à sa dérivée :

$$f'(N') = \frac{\log(2T)}{\log(2)N'} - \frac{t-2}{N-N'} - \log(T)$$

$f'(N') \geq 0$ si et seulement si :

$$\log(2T)(N-N') - (t-2)\log(2)N' - \log(T)\log(2)N'(N-N') \geq 0$$

En effet, le dénominateur de f', $N'(N-N')\log(2)$, est, par construction, toujours positif.

$f'(N') \geq 0$
$\Leftrightarrow N'^2 \log(T)\log(2) + N\log(2T) - N'\left(\log(2T) + (t-2)\log(2) + N\log(T)\log(2)\right) \geq 0$

Le discriminant Δ associé à cette équation est :

$$\Delta = \left((2-t)\log(2) - \log(2T) - N\log(T)\log(2)\right)^2 - 4N\log(T)\log(2T)\log(2)$$

Par hypothèse $0 \leq \tau < 1/2$, et $t \geq 1$, ce qui implique les deux encadrements suivants :

$$0 < \log(2T) \leq \log(2)$$
$$-\log(2) < \log(T) \leq 0$$

Il s'en déduit que :

$$4N\log(T)\log(2T)\log(2) \leq 0$$

Et donc pour tout t et tout τ dans l'intervalle $[0, 1/2[$ on a $\Delta \geq 0$. Ainsi, l'équation $f'(N') = 0$ admet deux solutions N'_1 et N'_2 :

$$N'_1 = \frac{\log(2T) + (t-2)\log(2) + N\log(T)\log(2) - \sqrt{\Delta}}{2\log(T)\log(2)}$$
$$N'_2 = \frac{\log(2T) + (t-2)\log(2) + N\log(T)\log(2) + \sqrt{\Delta}}{2\log(T)\log(2)}$$

De plus, $N'_2 < 0$, en effet, comme noté ci-dessus, $2\log(T) < 0$, d'où $N'_2 < 0$ est équivalent à :

$$\log(2T) + (t-2)\log(2) + N\log(T)\log(2) + \sqrt{\Delta} > 0 \tag{6.2}$$

Or $\Delta > \left((2-t)\log(2) - \log(2T) - N\log(T)\log(2)\right)^2$, par conséquent :

$$\sqrt{\Delta} > (2-t)\log(2) - \log(2T) - N\log(T)\log(2)$$

et en remplaçant $\sqrt{\Delta}$ dans l'équation (6.2), le numérateur de N'_2 est minoré par 0 :

$$\log(2T) + (t-2)\log(2) + N\log(T)\log(2) + \sqrt{\Delta} > 0$$

D'où $N'_2 < 0$, quelles que soient les valeurs de t, τ et N.

De la même façon, il se démontre que $N'_1 \geq 0$. Il s'ensuit que la fonction f définie sur $[0, N]$ est croissante sur $[0, N'_1]$ et décroissante sur $[N'_1, N]$ si $N'_1 < N$, sinon f est croissante sur tout son ensemble de définition. f atteint donc son maximum en $min(N, N'_1)$. Expérimentalement, il a été observé que $N'_1 < N$.

Le coût de la recherche d'un mot de code de poids t lorsque $p_1 = 1$ et $p_2 = 0$ est donc minimal pour :

$$\begin{cases} M' &= \frac{\log(N')}{\log(2)} - N' + N - 1 \\ N' &= min(N, N'_1) \end{cases}$$

□

D'après les simulations effectuées en Maple, le coût est minimal lorsque $p_2 = 0$, cependant p_1 peut être supérieur à 1. Un raisonnement similaire permettant de trouver les paramètres optimaux peut être appliqué pour d'autres valeurs de p_1.

Par exemple, sous l'hypothèse $p_1 = 2$ et $p_2 = 0$ le choix optimal de M' et N' est :

$$\begin{cases} M' &= \frac{2\log(N')}{\log(2)} - N' + N - 2 \\ N' &= min(N, N'_1) \end{cases}$$

avec

$$N'_1 = \frac{t - 2 + \log\left(\frac{1+(1-2\tau)^t}{2}\right)\left(\frac{4}{\log(2)} - N\right) - \sqrt{\Delta}}{2\log\left(\frac{1+(1-2\tau)^t}{2}\right)}$$

et

$$\Delta = \left(2 - t - \log\left(\frac{1+(1-2\tau)^t}{2}\right)\left(\frac{4}{\log(2)} - N\right)\right)^2 \\ -4\log\left(\frac{1+(1-2\tau)^t}{2}\right)\left(2N + \frac{4}{\log(2)}\log\left(\frac{1+(1-2\tau)^t}{2}\right)\right)$$

6.4.3 Exemples de paramètres optimaux (théoriques)

Afin de mieux comprendre le comportement des paramètres optimaux, la table 6.1 donne, pour différentes valeurs de N, les choix optimaux pour N' et M'. Ces paramètres sont obtenus en négligeant le coût du pivot de Gauss et en fixant $p_1 = 1$ et $p_2 = 0$.

Lorsque $\frac{t}{N} < \frac{1-(1-2\tau)^t}{2}$ il faut choisir $N' > M'$. Cette situation correspond aux colonnes en bleu de la table 6.1 et se justifie par le fait que, dans ce cas, la densité des éléments non-nuls est inversée entre les deux parties de l'équation recherchée (la partie droite est plus dense que la partie gauche).

Il peut être remarqué qu'une fois que $\frac{t}{N} < \frac{1-(1-2\tau)^t}{2}$ la valeur optimale de M' converge très rapidement. Les valeurs optimales ont également été recherchées de façon expérimentale, le coût du pivot de Gauss n'est alors pas négligé et le même phénomène de convergence de la valeur optimale de M' à partir d'un certain seuil (dépendant de N, M, t et τ) est observé.

De plus, il peut être observé que la valeur optimale de M' décroît lorsque le niveau de bruit augmente. Ceci se justifie par le choix de $p_2 = 0$. En effet, notons $i_1, i_2, \ldots, i_{M'}$ les colonnes choisies sur la partie identité de la matrice $\boldsymbol{\mathcal{H}}$. Une équation de parité $\boldsymbol{c}$ sera retrouvée si et seulement si les mots de code bruités se trouvant aux lignes $i_1, i_2, \ldots, i_{M'}$ ne contiennent pas d'erreur, ou un nombre pair d'erreurs, sur le support de $\boldsymbol{h}$. Lorsque le niveau de bruit est faible, la probabilité que les M' colonnes choisies permettent de retrouver $\boldsymbol{h}$ est élevée, un nombre important de colonnes peuvent être sélectionnées. Par contre, lorsque le niveau de bruit est plus important, cette probabilité décroît fortement, il est alors peu probable que les M' colonnes choisies soient telles que $\boldsymbol{h}$ puisse être retrouvé. D'où le choix (aléatoire) d'un petit nombre M' dans le cas très bruité.

$\tau = 0$								
N	100	200	500	1000	2000	5000	10000	20000
M'	92	183	453	901	1799	4489	8973	17939
N'	10	20	52	103	207	518	1035	2071
WF	24	24	25	25	25	25	26	26
$\tau = 0.0001$								
N	100	200	500	1000	2000	5000	10000	20000
M'	92	182	450	890	1746	4016	6142	7045
N'	10	21	54	116	261	993	3869	12967
WF	24	27	34	45	78	339	2441	50362
$\tau = 0.001$								
N	100	200	500	1000	2000	5000	10000	20000
M'	91	178	407	622	715	748	758	762
N'	11	26	98	384	1295	4262	9254	19251
WF	42	77	347	3×10^3	5×10^4	8×10^6	4×10^8	3×10^{10}
$\tau = 0.01$								
N	100	200	500	1000	2000	5000	10000	20000
M'	68	79	84	85	87	89	90	91
N'	36	127	424	923	1923	4923	9923	19923
WF	3×10^3	7×10^4	1×10^7	7×10^8	5×10^{10}	1×10^{13}	9×10^{14}	6×10^{16}

TABLE 6.1 – Valeurs théoriques optimales pour $t = 8$

6.5 Comparaison des différentes méthodes

La méthode de Dumer modifiée présentée ci-dessus permet de généraliser les méthodes de Valembois, Cluzeau-Finiasz et Gauss Randomisé. La table 6.2 représente les répartitions de poids des équations recherchées par ces différentes méthodes ainsi que les paramètres à choisir pour se rapporter à la méthode de Dumer modifiée.

En utilisant la proposition 6.2 déterminant le coût pour retrouver une équation de parité donnée, il se déduit le coût pour retrouver une équation de parité de poids t quelconque en utilisant la méthode de Dumer modifiée. Afin de comparer le coût de cette méthode avec celui des méthodes de Gauss Randomisé et de Cluzeau-Finiasz, le pivot de Gauss n'est plus négligé et est considéré comme cubique. De plus, il est supposé que le code recherché vérifie $N - K$ équations de parité de poids t et que les paramètres p_1 et p_2 sont respectivement fixés à 1 et 0. Ce choix sur les paramètres p_1 et p_2 n'est pas nécessairement le choix optimal mais permet tout de même de comparer les différentes méthodes. Sous ces hypothèses, le coût $\mathcal{C}_{DM}$ pour retrouver une équation de parité par la méthode de Dumer modifiée est donc :

$$C_{DM} = \frac{(M-\ell)^3 + N' + \frac{N'^2}{2^{N'+M'-N+1}}}{(N-K)\frac{N'^2}{2}\binom{N-N'}{t-2}\left(\frac{1+(1-2\tau)^t}{2}\right)^{M'}}\binom{N}{t}$$

La table 6.3 regroupe quelques évaluations des coûts des différentes méthodes. Le coût minimal est mis en évidence pour chacune des lignes. La méthode de Dumer modifiée ayant de nombreux paramètres, les valeurs présentées dans le tableau ne sont pas nécessairement

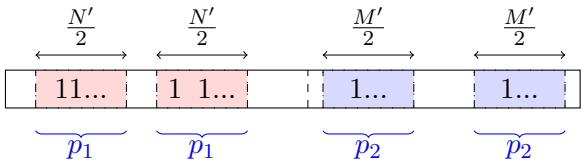
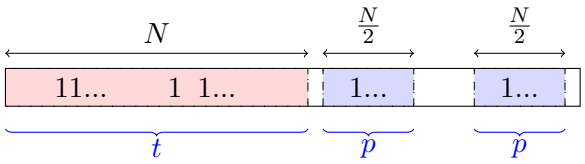
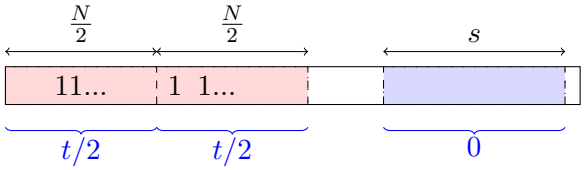
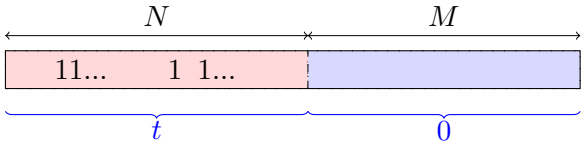
Méthode	Répartition de poids de $\boldsymbol{c}$	Paramètres
Dumer modifiée		
Valembois		$N' = N$, $p_1 = t/2$, $M' = M$ et $p_2 = p$
Cluzeau-Finiasz		$N' = N$, $p_1 = t/2$, $M' = s$ et $p_2 = 0$
Gauss Randomisé		$N' = 0$, $p_1 = 0$, $M' = M$ et $p_2 = 0$. De plus, $M - \ell = N$.

TABLE 6.2 – Généralisation des méthodes existantes

N	K	t	τ	C_{GR}	C_{CF}	C_{DM}
2 304	1 152	7	0	$\mathbf{1.1 \times 10^7}$	5.4×10^7	2.3×10^7
			0.001	1.0×10^{14}	6.7×10^7	$\mathbf{5.4 \times 10^7}$
			0.01	1.9×10^{75}	$\mathbf{4.8 \times 10^8}$	$\mathbf{4.8 \times 10^8}$
		20	0	$\mathbf{1.1 \times 10^7}$	2.7×10^{36}	5.4×10^7
			0.001	7.2×10^{26}	5.1×10^{36}	$\mathbf{5.7 \times 10^{17}}$
			0.01	7.8×10^{188}	9.1×10^{38}	$\mathbf{5.8 \times 10^{32}}$
16 200	14 400	27	0	$\mathbf{2.4 \times 10^9}$	1.1×10^{73}	1.8×10^{10}
			0.001	9.0×10^{196}	2.5×10^{73}	$\mathbf{9.1 \times 10^{46}}$
			0.01	5.5×10^{1669}	2.0×10^{76}	$\mathbf{2.7 \times 10^{66}}$
64 800	32 400	7	0	$\mathbf{8.4 \times 10^9}$	1.4×10^{16}	1.3×10^{10}

TABLE 6.3 – Comparaison des coûts des méthodes de Gauss Randomisé, de Cluzeau-Finiasz et de Dumer Modifiée

les valeurs optimales. Un seul paramètre est à fixer pour la méthode de Cluzeau-Finiasz, noté s dans la sous-section consacrée à cette méthode, il a été fixé à 32 comme préconisé dans [24].

Lorsque les données ne sont pas bruitées la méthode de Gauss Randomisé est la moins coûteuse mais dans ce cas la méthode de Dumer modifiée a tout de même un coût relativement proche, ce qui n'est pas le cas de la méthode de Cluzeau-Finiasz si le poids des équations est important ou si la longueur du code est importante. Lorsque le bruit augmente les coûts des différentes méthodes varient plus nettement, ceux des méthodes de Cluzeau-Finiasz et de Dumer modifiée se rejoignant pour des niveaux de bruit très élevés. Cependant, en utilisant les équations qui se déduisent directement de l'étape du pivot de Gauss, le coût de la méthode de Dumer modifiée peut être inférieur ou égal aux coût des deux autres méthodes quel que soit le niveau de bruit considéré.

Pour comparer ces différentes méthodes le coût est important mais la quantité de données nécessaire également. La méthode de Gauss Randomisé nécessite un ensemble de plus de N mots, alors que les méthodes de Cluzeau-Finiasz et celle de Dumer modifiée utilisent beaucoup moins de mots. Par construction, le nombre de mots utilisés par la méthode de Dumer modifiée est forcément inférieur à N il est même généralement très largement inférieur à N comme dans les tests du chapitre suivant.

Chapitre 7

Reconnaissance des codes LDPC : Résultats expérimentaux

La méthode présentée dans le chapitre précédent pour retrouver des équations de parité de (petit) poids t d'un code inconnu $\mathcal{C}$ à partir d'un ensemble de M mots de code bruités est testée afin d'évaluer ses capacités et ses limites. Dans ce chapitre sont présentés et commentés quelques résultats expérimentaux. Ces résultats concernent, dans un premier temps, le cas non bruité, puis, dans un second temps, le cas bruité. Ils ont tous été obtenus en modifiant l'implémentation, faite par Gregory Landais [80], de la méthode de Dumer afin d'obtenir la méthode modifiée.

7.1 Cas non bruité

Supposons qu'un ensemble de M mots de code (non bruités) est donné, le but est de reconstruire la matrice de parité définie par un ensemble d'équations de parité de petit poids.

7.1.1 Exemple : codes quasi-cycliques après permutation des colonnes

La table 7.1 présente les résultats obtenus pour les codes LDPC binaires de la norme DVB-S2 [42]. Pour rappel, ces codes possèdent une structure convolutive et sont quasi-cycliques après permutation des colonnes de la matrice de parité. Cette structure très précise n'est pas utilisée pour la reconnaissance du code. Pour un code normé de longueur N, de dimension K et défini par le polynôme $\rho(x)$, le temps pris pour reconstruire entièrement la matrice de parité est donc identique à celui nécessaire pour reconstruire un code LDPC quelconque de longueur N, de dimension K et défini par le même polynôme $\rho(x)$.

Les codes LDPC de cette norme font partie des codes normés les plus longs, en effet dans ce cas $N = 16\,200$ ou $64\,800$. Tous les codes de longueur $16\,200$ ont été reconstruits, pour ceux de longueur $64\,800$, les temps de calculs deviennent importants et reconstruire entièrement la matrice de parité sans hypothèse sur la structure du code ne semble pas très adapté, en particulier lorsque le rendement est important. Cependant, il peut être souligné que la reconstruction totale de la matrice de parité, qui nécessite donc de retrouver $N - K$ mots différents de petit poids, est efficace sur les codes de longueur $N = 16\,200$ même si les équations de parité sont de poids relativement élevées (jusqu'à $t = 27$ dans la table 7.1).

N	$N-K$	poids max.	M	temps de recherche
16 200	12 960	4	800	30 sec.
16 200	10 800	5	1 900	60 sec.
16 200	9 720	6	2 800	140 sec.
16 200	9 000	7	2 300	2 000 sec.
16 200	6 480	11	7 000	1 800 sec.
16 200	5 400	10	6 400	1 200 sec.
16 200	4 320	13	6 900	5 000 sec.
16 200	3 600	13	8 000	3 000 sec.
16 200	2 880	19	9 900	6 000 sec.
16 200	1 800	27	12 100	5 000 sec.
64 800	48 600	4	1 600	600 sec.
64 800	43 200	5	4 300	3 200 sec.
64 800	38 880	6	9 000	15 000 sec.
64 800	32 400	7	13 000	30 000 sec.

TABLE 7.1 – Reconnaissance des codes LDPC de la norme DVB-S2

De plus, la méthode permettant de retrouver les équations de parité est itérative, et chaque itération est indépendante des précédentes et des suivantes. Le temps de recherche peut donc être divisé par simple parallélisation.

Influence du polynôme $\rho(x)$

Le polynôme $\rho(x)$ définit l'uniformité du poids des lignes de la matrice de parité : $\rho(x) = \sum_i \rho_i x^i$ où ρ_i est égal à la proportion de lignes de poids i dans la matrice de parité.

Les deux lignes de la table 7.1 pour lesquelles le poids maximal est 13 permet de mettre en évidence l'impact du polynôme $\rho(x)$ sur la reconnaissance du code LDPC. Pour ces codes $\rho(x)$ est défini par :

— pour $N-K = 4\,320$: $\rho(x) = 0.08x^9 + 0.25x^{10} + 0.33x^{11} + 0.25x^{12} + 0.08x^{13}$

— pour $N-K = 3\,600$: $\rho(x) = 0.1x^{11} + 0.3x^{12} + 0.6x^{13}$

Il faut environ 1.16 secondes par équation retrouvée pour le premier code contre 0.83 secondes pour le second.

Pour le premier de ces codes, la proportion d'équations de parité de poids maximal est inférieure à celle du second code (0.08 contre 0.6) alors que le temps de reconstruction est supérieur, ce qui semble contre-intuitif. En réalité, ce sont les équations de parité de poids inférieur qui ralentissent la reconnaissance. En effet, ces équations sont retrouvées de nombreuses fois et le coût de la vérification qu'il s'agit bien d'une équation de parité (c'est-à-dire vérifier que son poids est inférieur à $w-2p_1-2p_2$ sur les premiers indices) n'est pas négligeable.

Le même phénomène se produit pour les codes de longueur 16 200 et ayant des équations de parité de poids maximal 7 et 19. En effet, l'un satisfait des équations de parité de poids allant de 4 à 7 et l'autre de poids compris entre 15 et 19. Un code vérifiant uniquement des équations de poids 7 ou 19 serait donc reconnu plus rapidement que ceux-ci.

Les équations de parité des codes de longueur $N = 64\,800$ sont toutes de poids maximal t

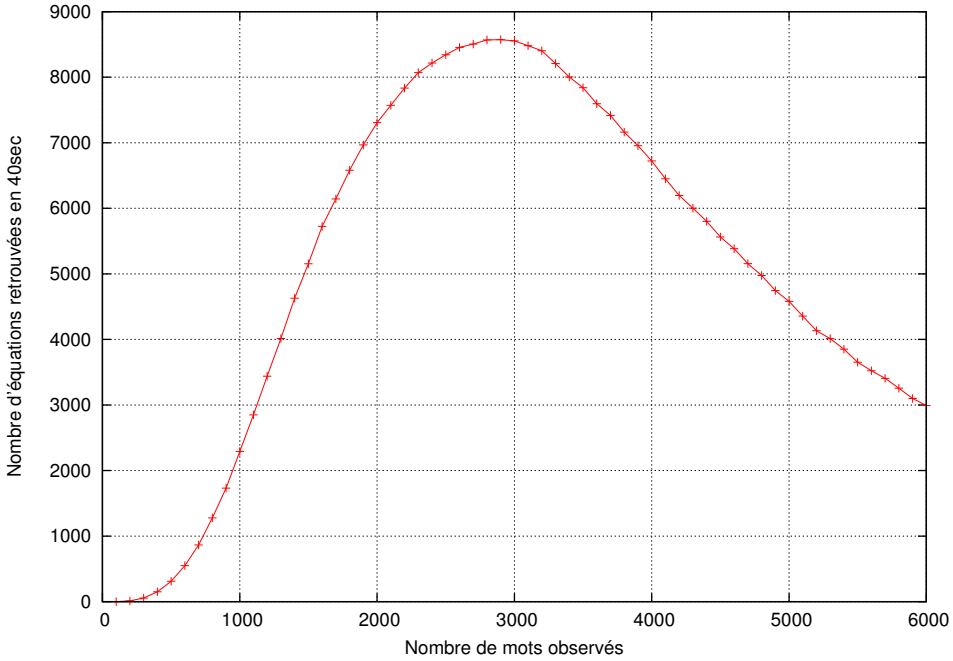


FIGURE 7.1 – Influence du nombre de mots utilisés sur le nombre d'équations de parité retrouvées en un temps donné

sauf une qui est de poids $t-1$. Les temps de reconnaissance ne sont donc pas influencés par le polynôme $\rho(x)$ mais directement par la valeur de t.

Choix de M

La valeur M indiquée dans la table 7.1 correspond au nombre de mots optimal, c'est-à dire à la valeur permettant de minimiser le temps de la reconnaissance (valeur estimée de façon pratique). Plus le poids des équations augmente, plus M croît. De plus, ces temps sont atteints pour $p_1 = 1$ et $p_2 = 0$, qui sont les paramètres les mieux adaptés pour ces codes. Afin de trouver une équation de parité donnée, il faut qu'un seul de ces indices appartienne aux $N'/2$ premières positions choisies et un seul aux $N'/2$ autres positions choisies. Plus t augmente plus il est difficile de faire un tel choix (aléatoirement), il faut donc que $N'/2$ décroisse ce qui se fait en augmentant M. Ainsi plus t est faible, plus le nombre de mots nécessaires est minime.

Le code normé de longueur $N = 16\,200$ et de dimension $K = 6\,480$ satisfait $9\,719$ équations de parité de poids 6 et 1 de poids 5. La courbe de la figure 7.1 représente le nombre d'équations de parité retrouvées pour ce code en un temps donné en fonction du nombre de mots M utilisés. Elle possède toujours les mêmes variations, c'est-à-dire une croissance très rapide jusqu'à la valeur de M optimale puis une diminution plus lente. Le choix du nombre de mots utilisé est donc très important, il est inutile d'avoir plus de mots que nécessaire, cela n'accélère pas la reconstruction.

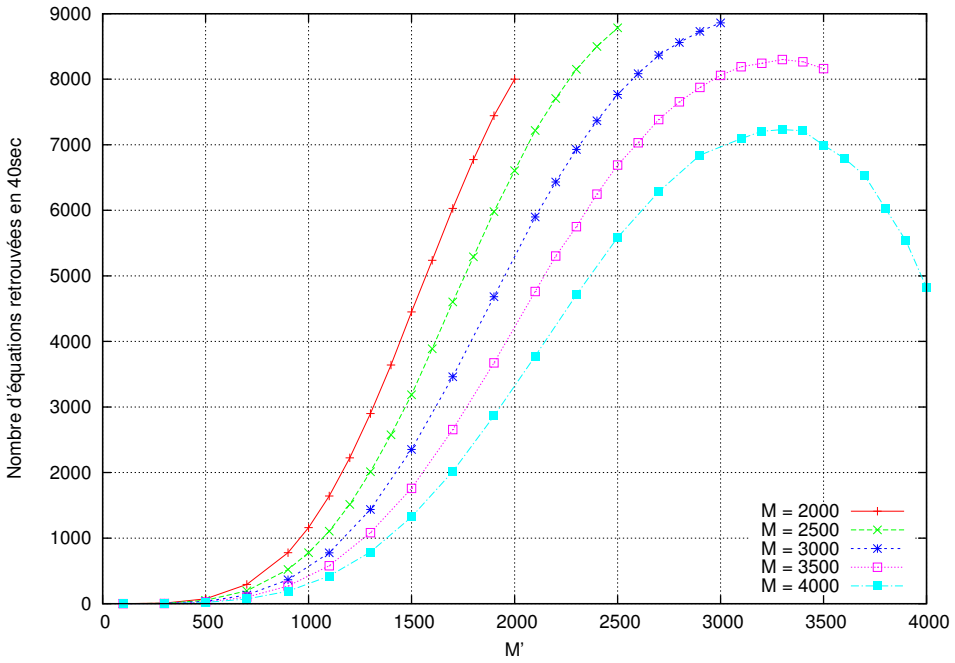


FIGURE 7.2 – Influence de M' sur le nombre d'équations de parité retrouvées en un temps donné

Choix de M' et N'

Nous nous intéressons ici au choix de la variable M' et à son impact sur le temps de recherche des équations de parité. La figure 7.2 représente le nombre d'équations de parité retrouvées en 40 secondes en fonction de M', et ceci pour des nombres M de mots observés différents. Rappelons que, par construction, il est impossible que $M' > M$ et que les paramètres M' et N' sont liés, l'influence de N' se déduit donc de celle de M'. En ajoutant quelques courbes à cette figure, il peut être déduit que les valeurs optimales de M et M' sont 2 800.

Choix le ℓ

Le paramètre ℓ doit également être choisi de façon optimale et très précisément, sa valeur influence directement les performances de l'algorithme de recherche des mots de petit poids. S'il est trop petit les listes $\mathcal{L}_1$ et $\mathcal{L}_2$ génèrent de nombreuses collisions, pour chacune d'elles il faut tester le poids total du vecteur correspondant ce qui prend du temps. Il faut mieux tester seulement les vecteurs qui ont une forte probabilité d'être de poids inférieur ou égal à w. A l'inverse si ℓ est trop grand, le coût de la création et de la recherche des collisions entre les deux listes augmente fortement. Il n'y aura que peu de collisions à tester et chacune d'elle aura une grande probabilité de conduire à un mot de petit poids, mais le coût de la gestion des listes n'est tout de même pas compensé.

Toujours pour le code LDPC normé de longueur $N = 16\,200$ et de dimension $K = 6\,480$, la figure 7.3 représente l'influence de ℓ sur le nombre de mots de petit poids retrouvés en un temps donné, pour les paramètres M et M' optimaux choisis précédemment (égaux à 2 800).

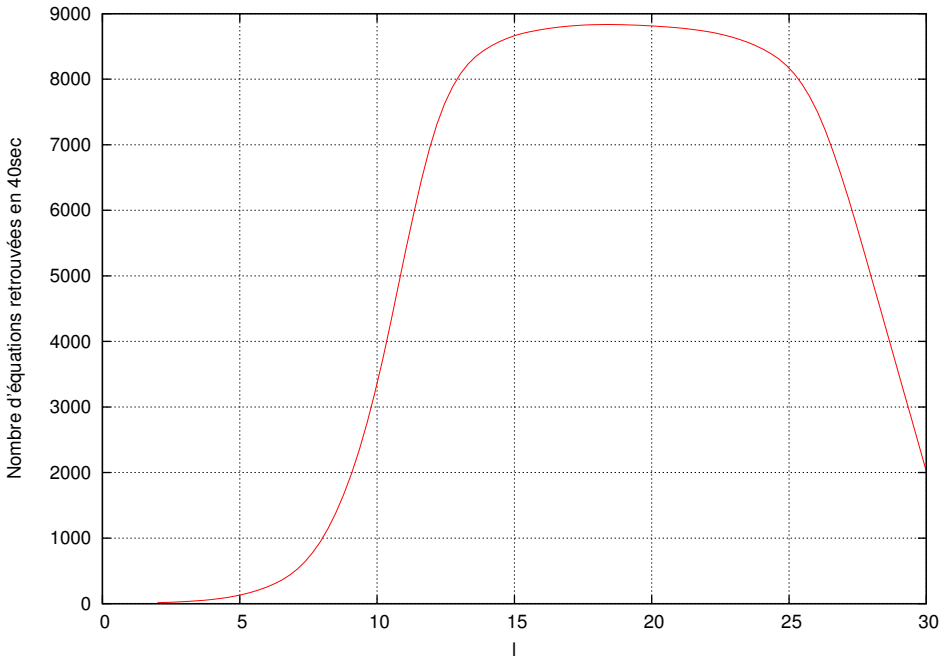


FIGURE 7.3 – Influence de ℓ sur le nombre d'équations de parité retrouvées en un temps donné

7.1.2 Exemple : codes quasi-cycliques

D'autres tests ont été effectués dans le cas non bruité sur des codes LDPC quasi-cycliques (codes des normes IEEE 802.16* et 802.22). Là non plus, la structure de la matrice de parité n'a pas été utilisée lors de sa reconstruction. Ces codes sont de longueur comprise entre 384 et 2 304, ils sont tous reconstruits extrêmement rapidement comme le montre la table 7.2. En effet, pour tout ces codes il faut moins de 8 secondes pour retrouver toutes les équations de parité de petit poids, et donc reconstruire la matrice de parité souhaitée. C'est également le cas pour des équations de parité de poids 20, ce qui est élevé par rapport à la longueur des codes considérés.

La construction de ces codes est facilement extensible à des longueurs plus importantes, des tests ont donc été effectués sur des longueurs non normées afin de mettre en évidence l'influence de la longueur sur les paramètres optimaux et sur le temps de la reconnaissance. La figure 7.4 représente le nombre de mots M optimal en fonction de la longueur N du code, pour des codes LDPC ayant des équations de poids 6, 10 ou 14. Ces valeurs ont été obtenues de façon expérimentale. Plus le poids des équations recherchées est important plus la quantité optimale de données est importante, cette quantité croit quasiment linéairement avec la longueur du code considéré. Quant à la figure 7.5 elle représente le temps nécessaire pour retrouver toutes les équations de parité en fonction de la longueur du code, ce code étant de rendement 1/2 et vérifiant uniquement des équations de parité de poids 6.

N	$N-K$	poids max.	M	temps de recherche
384	192	7	100	1 sec.
864	432	7	300	1 sec.
1248	624	7	400	2 sec.
1728	864	7	600	2 sec.
2304	1152	7	700	4 sec.
384	128	10	200	1 sec.
864	288	10	400	2 sec.
1248	416	10	700	2 sec.
1728	576	10	800	3 sec.
2304	768	10	1100	5 sec.
384	95	15	200	2 sec.
864	216	15	600	2 sec.
1248	312	15	800	2 sec.
1728	432	15	1100	4 sec.
2304	576	15	1400	7 sec.
384	64	20	300	1 sec.
864	144	20	700	2 sec.
1248	208	20	900	3 sec.
1728	288	20	1300	4 sec.
2304	384	20	1700	8 sec.

TABLE 7.2 – Reconnaissance des codes LDPC des normes 802.16* et 802.22

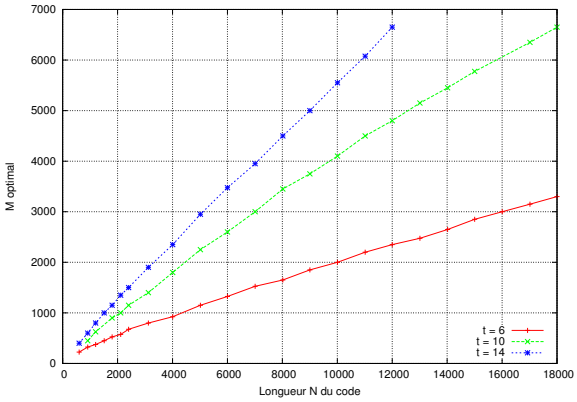


FIGURE 7.4 – Influence de N sur la valeur optimale de M

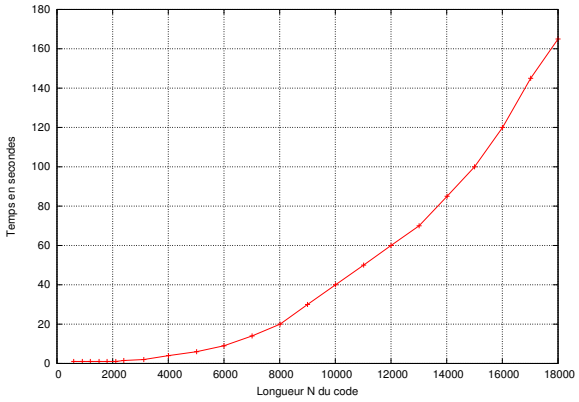


FIGURE 7.5 – Influence de N sur le temps pour retrouver toutes les équations

7.2 Cas bruité

De nombreux tests ont été effectués dans le cas non bruité, mais le cas bruité a également été traité, cette section lui est consacrée. Comme précédemment, ces tests ont concernés deux types de codes LDPC normés : ceux de la norme DVB-S2 et ceux des normes IEEE 802.16* et 802.22.

Les structures très fortes des matrices de parité recherchées ne sont pas prises en compte. La méthode de Dumer modifiée est donc appliquée le temps nécessaire pour retrouver toutes les équations de parité sans utiliser la présence de cette structure.

La recherche aveugle des équations de parité prend du temps à cause de la répartition de poids imposée au mot de code $\boldsymbol{c}$. De plus, le poids du vecteur recherché est plus important que dans le cas non bruité, ce qui augmente également le temps nécessaire.

Une des propriétés des codes LDPC est leur décodage linéaire. Ce décodage peut être effectué sans disposer de l'ensemble des équations de parité, c'est-à-dire en ayant seulement une sous-matrice de la matrice de parité. Appliquer régulièrement une étape de correction sur les mots bruités observés permet de réduire le niveau de bruit et d'accélérer la reconstruction.

Sur la figure 7.6 est tracé le nombre d'équations de parité retrouvées en fonction du temps consacré à leur recherche pour le code LDPC de longueur 16 200 et de rendement 2/5 de la norme DVB-S2. Ce code vérifie 9 719 équations de poids 6 et 1 équation de poids 5, les mots observés ont été bruités par un canal binaire symétrique de probabilité $\tau = 10^{-4}$. La reconstruction de la matrice de parité a été tentée à partir de deux ensembles de mots de code bruités, l'un en contenant $M = 2\,500$, l'autre $M = 2\,800$.

Afin de mettre en évidence l'apport de corrections intermédiaires, pour chacun des ensembles de mots observés, il est tracé la courbe obtenue sans correction lors de la recherche ainsi que la courbe avec une correction des mots bruités toutes les 270 secondes. Le temps de cette correction est négligé, et n'est donc pas pris en compte sur cette figure.
Le gain apporté par ces corrections est significatif, par exemple, avec 2 800 mots de code bruités, cela permet de passer de 5 000 équations de parité retrouvées à plus de 7 700 lors d'une recherche effectuée durant 10 000 secondes.

Pour cet exemple le choix du nombre de mots de code utilisés n'est pas aléatoire, en effet, $M = 2\,800$ est le choix optimal pour retrouver ce code dans le cas non-bruité. Cependant, $M = 2\,500$ est la valeur optimale pour reconstruire ce code lorsque le niveau de bruit est de 10^{-4}. Cela peut sembler contre-intuitif, mais lorsque le bruit augmente, le nombre de mots à observer afin de minimiser le temps de la reconstruction diminue. En effet, cela permet de maximiser la probabilité de retrouver une équation donnée ainsi que de réduire le poids des vecteurs recherchés. Ce comportement est également présent dans l'analyse théorique du choix optimal des paramètres.

Pour un niveau de bruit plus important le gain apporté par des corrections partielles permet de réduire de façon encore plus significative le temps de la reconstruction. La figure 7.7 représente, comme précédemment, le nombre d'équations retrouvées en fonction du temps de recherche, avec et sans étapes de corrections intermédiaires, toujours pour le code normé de longueur $N = 16\,200$ et de rendement $R = 2/5$. Pour cette reconstruction, un ensemble de 1 000 mots est observé en sortie d'un canal binaire symétrique de probabilité d'erreur

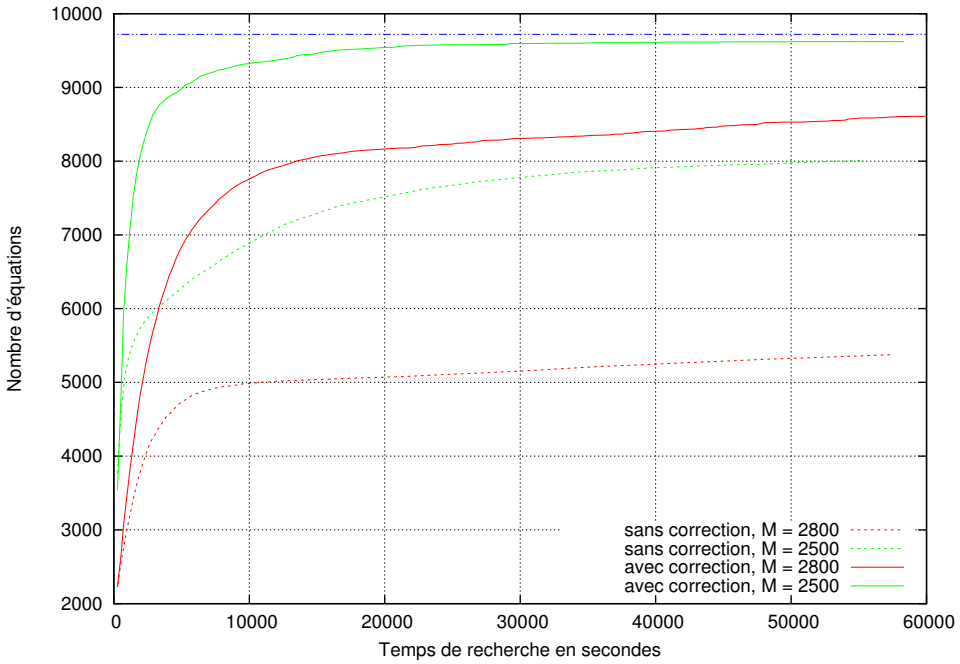


FIGURE 7.6 – Évolution du nombre d'équations retrouvées en fonction du temps et en effectuant ou non des corrections intermédiaires

10^{-3}. Sur cette figure, comme sur la figure 7.6, il semble très difficile de retrouver les toutes dernières équations mais les équations retrouvées jusque là sont suffisantes pour corriger les mots bruités observés. Il est donc possible de se ramener au cas non bruité en appliquant une simple correction à un ensemble de mots plus conséquent afin d'utiliser des paramètres proches des valeurs optimales. Pour cet exemple, il suffirait alors de 140 secondes supplémentaires pour retrouver l'ensemble des équations de parité à partir d'un ensemble de 2 800 mots parfaitement corrigés.

La méthode a également été testée sur des codes LDPC quasi-cycliques normés, ceux des normes IEEE 802.16* et 802.22. La table 7.3 regroupe les temps nécessaires afin de retrouver l'ensemble des équations de parité de certains codes appartenant à ces normes. Lors de ces recherches, des corrections intermédiaires ont été effectuées ce qui permet une reconstruction plus rapide. La figure 7.8 représente le nombre d'équations de parité retrouvées en fonction du temps pour le code normé de longueur $N = 1\,728$ et de rendement $R = 2/3$. Ce code satisfait des équations de parité de poids 10, 700 mots bruités par un canal binaire symétrique de probabilité d'erreur $\tau = 10^{-3}$ sont utilisés pour cette reconnaissance. Les corrections partielles permettent de retrouver toutes les équations en 2h 15 minutes alors que sans correction intermédiaire moins de 50% des équations sont retrouvées dans le même temps.

Que les données observées soient bruitées ou non, de nombreuses équations de parité sont retrouvées très rapidement avec la méthode de Dumer modifiée. Cependant, les dernières équations de parité à retrouver sont toujours les plus coûteuses, ceci correspond au problème du collectionneur de coupons [50]. Utiliser la structure du code LDPC devrait permettre de réduire le temps nécessaire pour reconstruire le code recherché.

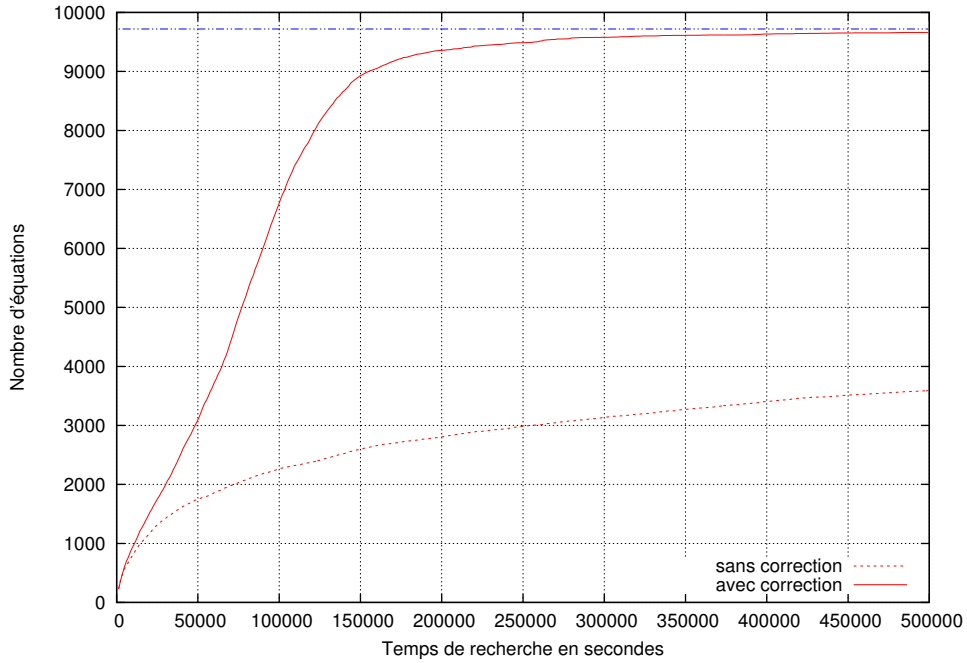


FIGURE 7.7 – Évolution du nombre d'équations retrouvées en fonction du temps et en effectuant ou non des corrections intermédiaires

τ	N	$N-K$	poids max.	M	temps de recherche
10^{-3}	384	192	7	100	3 sec.
10^{-3}	864	432	7	200	9 sec.
10^{-3}	1248	624	7	200	60 sec.
10^{-3}	1728	864	7	300	72 sec.
10^{-3}	2304	1152	7	600	300 sec.
10^{-2}	384	192	7	100	66 sec.
10^{-2}	864	432	7	100	1 620 sec.
10^{-2}	1248	624	7	150	1 900 sec.
10^{-2}	1728	864	7	200	3 500 sec.
10^{-2}	2304	1152	7	500	11 000 sec.

TABLE 7.3 – Reconnaissance des codes LDPC des normes 802.16* et 802.22 à partir de mots bruités par un canal binaire symétrique de probabilité d'erreur τ

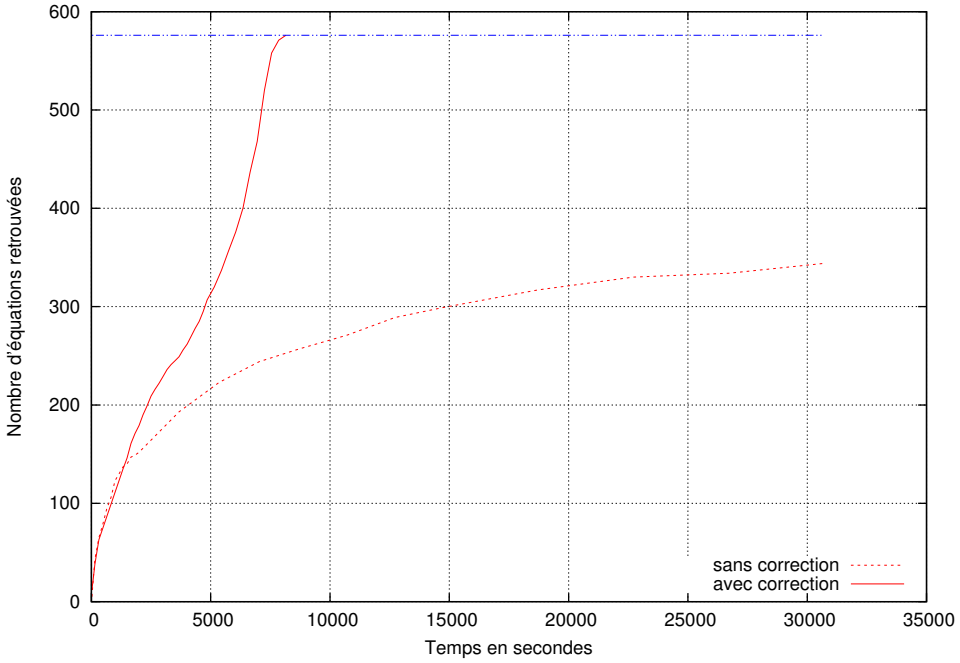


FIGURE 7.8 – Évolution du nombre d'équations retrouvées en fonction du temps et en effectuant ou non des corrections intermédiaires

7.3 Idées pour utiliser les structures des codes LDPC normés

L'étude des normes de télécommunication utilisant des codes LDPC a permis de mettre en évidence que tous ces codes sont extrêmement structurés. Or cette structure n'est pas utilisée lors de la reconstruction dans la méthode de Dumer modifiée. De plus, le nombre de structures utilisées en pratique est limité, il s'agit principalement de codes quasi-cycliques ou quasi-cyclique après permutation des colonnes. Il est donc intéressant de reconnaître la structure du code recherché afin d'accélérer la reconstruction de la matrice de parité. La recherche des dernières équations de parité est la partie la plus coûteuse de la reconstruction, mais retrouver environ $90 \sim 95\%$ des équations de parité peut être effectué très rapidement par la méthode de Dumer modifiée. Grâce à ces équations, la structure du code peut être déduite puis utilisée pour retrouver les dernières équations de parité manquantes. Deux idées pour utiliser cette structure sont présentées dans cette section, l'une est visuelle, l'autre analytique.

7.3.1 Représentation de la sous-matrice de parité

En représentant visuellement les équations de parité retrouvées jusque là la structure du code LDPC peut être déduite, en particulier dans le cas des codes LDPC quasi-cycliques. Une fois cette structure retrouvée il suffit de déduire les équations manquantes puis de compter le nombre de mots de code bruités vérifiant ces équations. Si ce nombre est supérieur à un seuil alors l'équation est ajoutée à la liste des équations retrouvées, sinon elle est rejetée. Ce seuil dépend du poids t des équations de parité recherchée, du nombre de mots M ainsi que du niveau de bruit introduit par le canal. Il peut être fixé autour de $(1-(1-2\tau)^t)M$, c'est-à-dire deux fois le nombre moyen de mots bruités ne vérifiant pas une équation de parité donnée de

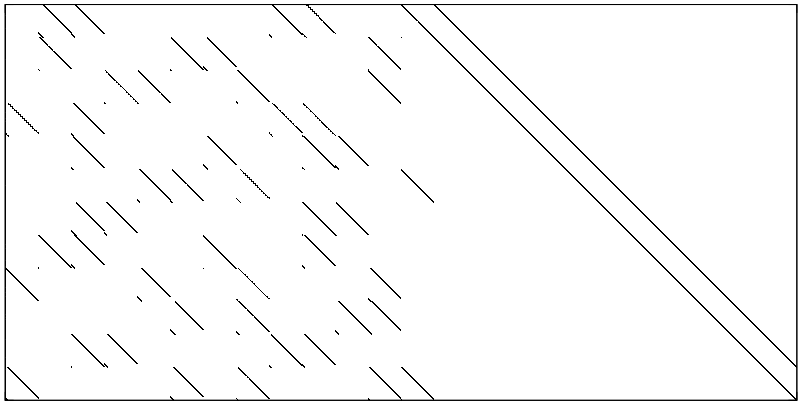


FIGURE 7.9 – Matrice obtenue avec 6 300 équations de parité (soit 84% des équations)

poids t.

Par exemple, pour un code LDPC quasi-cyclique de longueur $N = 15\,000$ et de rendement $1/2$, il faut 100 secondes pour retrouver ses 7 500 équations de parité de poids 6. Mais en seulement 30 secondes, plus de 6 300 équations sont retrouvées. En triant simplement ces équations en fonction de leur indice maximal, la matrice obtenue est représentée sur la figure 7.9. De cette matrice, il est facile de déduire la taille des sous-matrices cycliques. Puis, grâce à la double diagonale de déduire l'ensemble des équations manquantes. En effet, la matrice partielle obtenue possède des "trous" dans sa double diagonale sur les lignes correspondantes à des équations manquantes. Des lignes précédant et suivant une équation manquante il se déduit l'équation recherchée. Il faut ensuite la tester sur les mots de code bruités comme expliqué précédemment.

Un autre exemple de la représentation des équations de parité retrouvées jusqu'ici est donné par la figure 7.10. Cette figure représente la matrice obtenue après 300 secondes de recherche pour la reconstruction du code LDPC normé de longueur $N = 1\,728$ et de rendement $1/2$ connaissant un ensemble de 200 mots de code bruités par un canal binaire symétrique de probabilité d'erreur $\tau = 10^{-2}$. De cette matrice contenant 65% des équations il se déduit facilement les équations manquantes alors que la méthode de Dumer modifiée nécessite 2 700 secondes supplémentaires pour retrouver toutes ces équations manquantes. Sur cette même figure il est également mis en avant le fait que les équations de parité de plus petit poids sont retrouvées plus rapidement que les autres. En effet, les équations de poids 6 sont quasiment toutes connues à cet instant de la reconnaissance ce qui n'est pas le cas des équations de parité de poids 7.

En procédant de cette façon, il suffit de posséder une partie des équations de parité pour déduire les dernières, qui sont les plus coûteuses à retrouver par la méthode de Dumer modifiée, en particulier dans le cas bruité. Cette méthode s'applique particulièrement aux codes LDPC quasi-cycliques ou avec une structure convolutive mais peut atteindre ses limites avec d'autres types de codes, tels que celui de la norme IEEE 802.3 [71] (qui n'est ni quasi-cyclique ni avec une structure convolutive).

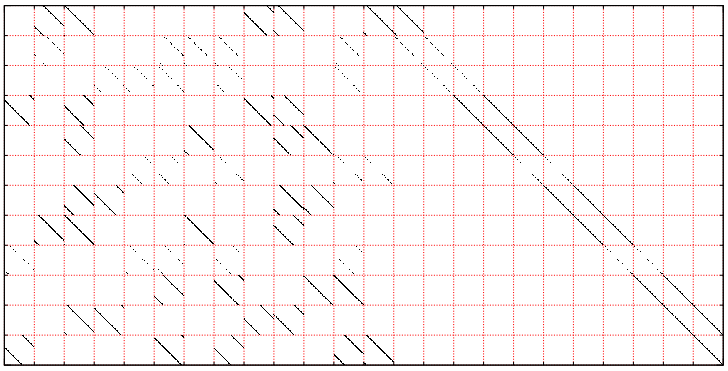


FIGURE 7.10 – Matrice obtenue avec 560 équations de parité (soit 65% des équations)

7.3.2 Étude de l'histogramme

Une autre idée pour améliorer la reconnaissance des codes LDPC structurés est de tracer l'histogramme représentant, pour chaque indice, le nombre de fois où il intervient dans une équation de parité retrouvée. Grâce à cet histogramme, il est possible de déduire un ensemble d'indices dont on est sûr que toutes les équations de parité les faisant intervenir sont déjà retrouvées. Ces indices n'intervenant pas dans les équations manquantes, les mots de code bruités sont poinçonnés sur ces positions. La méthode de Dumer modifiée est de nouveau appliquée mais uniquement sur les mots de code bruités poinçonnés. La longueur de ces mots est, par construction, inférieure à N, la recherche des équations manquantes est alors moins coûteuse et donc plus rapide.

Cette méthode est particulièrement intéressante pour les codes LDPC ayant des matrices de parité dont les poids des colonnes est constant ou constant par bloc. C'est par exemple le cas des codes LDPC des normes DVB ainsi que des codes quasi-cycliques mais également des codes qui sont moins structurés tels que celui de la norme IEEE 802.3.

L'exemple de la reconnaissance du code normé de longueur $N = 16\,200$, vérifiant $9\,719$ équations de parité de poids 6 et une seule équation de parité de poids 5 à partir d'un ensemble de $2\,500$ mots de code bruités par un canal binaire symétrique de probabilité $\tau = 10^{-4}$ est repris ici en utilisant un histogramme. Sur la figure 7.6, il est mis en évidence la difficulté de retrouver les dernières équations de parité. Alors que $9\,600$ équations sont retrouvées en $15\,000$ secondes il faut plus de $40\,000$ secondes supplémentaires pour retrouver les 120 équations de parité manquantes. L'histogramme obtenu à partir de ces $9\,600$ équations de parité est tracé sur la figure 7.11.

À partir de cet histogramme il se déduit que les $2\,160$ premiers indices interviennent dans 12 équations de parité, les indices compris entre $2\,160$ et $6\,480$ appartiennent eux à 3 équations de parité, enfin les $9\,720$ derniers indices doivent être impliqués dans 2 équations de parité (ils correspondent à la double diagonale présente sur la partie droite des matrices de parité des normes DVB). Seuls les indices n'atteignant par la valeurs souhaitée sont conservés, ils sont au nombre de 633. Les mots de codes bruités sont poinçonnés et de longueur 633. Seuls les 300 premiers mots bruités poinçonnés sont utilisés lors de la nouvelle recherche des mots de petits poids. Sur cet exemple, les 120 équations de parité manquantes sont trouvées en moins de 60 secondes. Le temps total de la reconstruction de ce code LDPC passe donc de

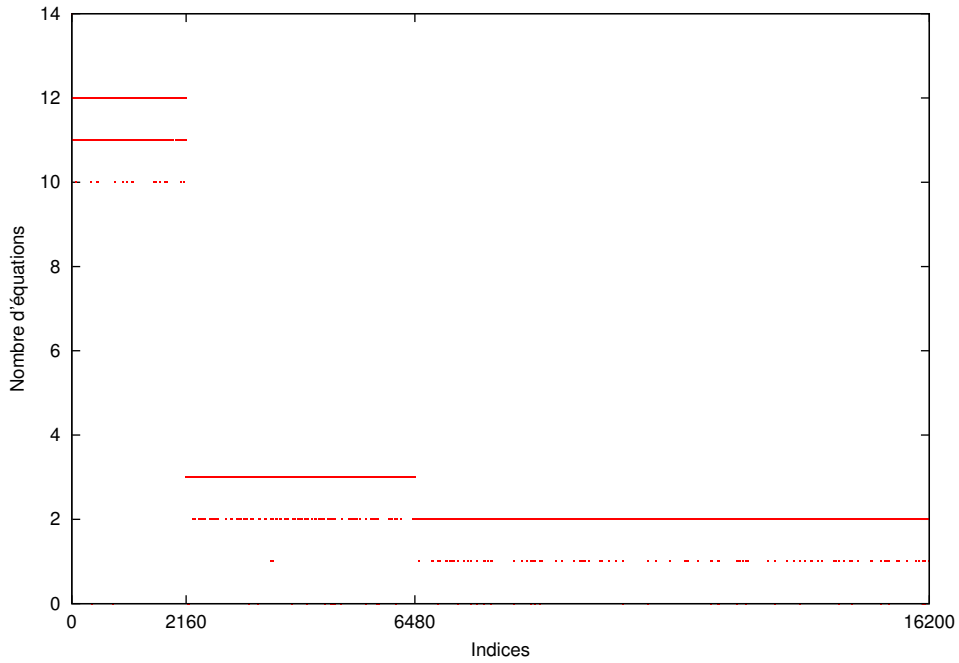


FIGURE 7.11 – Nombre d'équations de parité dans lesquelle interviennent les indices des mots de code

plus de 55 000 secondes à environ 15 060 secondes.

7.4 Conclusion

Dans ce chapitre, nous avons présenté quelques uns des tests effectués. La méthode de recherche des équations de parité de petit poids a permis de reconnaître les codes LDPC normés, dans le cas non bruités comme dans le cas bruité et même lorsque le poids des équations de parité de ces codes sont relativement élevés ou que les codes sont de longueur importante. Les dernières équations de parité sont les plus coûteuses à retrouver, c'est à ce moment qu'utiliser la structure des codes LDPC normés est intéressant afin de réduire le temps de la reconnaissance, mais n'est pas indispensable. En effet, la méthode proposée est itérative et donc parallélisable ce qui réduit d'autant le temps nécessaire à la reconstruction.

Comme montré par ces résultats, le choix des paramètres est un point délicat et peut avoir des conséquences importantes sur le temps pris par la reconnaissance. D'où l'intérêt de choisir des paramètres proches des valeurs optimales. Parmi ces paramètres, intervient le nombre de mots de code bruités observés utilisés pour reconnaître le code LDPC. Il a été mis en évidence dans ces différents tests la faible quantité de données nécessaire pour retrouver l'ensemble des équations de parité. De plus, il est inutile d'avoir plus de mots, la reconnaissance n'en étant pas accélérée.

Cette méthode de recherche des équations de petit poids s'applique particulièrement aux codes LDPC mais pas uniquement, comme nous allons le voir dans le chapitre suivant, elle est également utilisée pour la reconnaissance des codes convolutifs entrelacés.

Chapitre 8

Reconnaissance des codes convolutifs entrelacés

8.1 Problème et hypothèses

Ce chapitre est consacré, comme son nom l'indique, à la reconnaissance des codes convolutifs entrelacés. Plus précisément, les étapes permettant de passer de l'information au mot reçu sont représentées sur la figure 8.1 où le code $\mathcal{C}$ est un code de longueur N et de dimension K. Le vecteur $\boldsymbol{u}$ est donc de taille K et les autres vecteurs sont tous de longueur N. La reconstruction d'un code entrelacé est l'objet de différents articles. Par exemple dans [55, 73–75, 92] il est proposé des méthodes permettant de retrouver certains paramètres de l'entrelaceur tels que sa longueur N, ou une estimation du rendement du code utilisé. Cependant, dans ces articles, l'entrelaceur n'est pas réellement reconstruit, alors que dans les articles [55, 73–75, 92] l'entrelaceur utilisé est reconstruit mais en faisant de fortes hypothèses sur sa structure. En effet, les entrelaceurs considérés sont convolutifs [55, 74, 75, 92] ou hélicoïdaux [73]. De plus, quasiment toutes ces méthodes sont très sensibles au bruit ajouté par le canal, certaines s'appliquant uniquement sur des mots non bruités.

Une nouvelle méthode de reconstruction de l'entrelaceur est présentée ici. Elle n'utilise pas la structure de l'entrelaceur qui est supposé être une permutation aléatoire. Cette méthode fait l'objet d'un article à la conférence ISIT 2015 [128] ainsi que d'une version longue [129] sur le site d'archives en ligne ArXiv.

Pour cette méthode, les hypothèses sont les suivantes :
— Le code $\mathcal{C}$ est un code convolutif de type (n, k) inconnu.
— L'entrelaceur π est de longueur N connue.
— L'entrelaceur π est choisi de manière uniforme parmi les $N!$ possibles.
— Le canal utilisé est sans mémoire, son type ainsi que sa probabilité d'erreur sont connus.

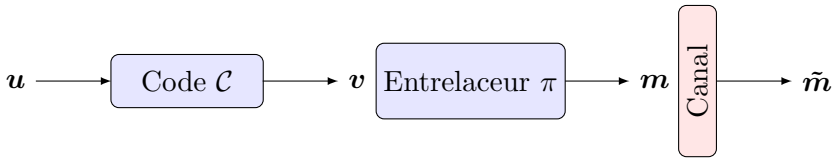


Figure 8.1 – Schéma d'encodage

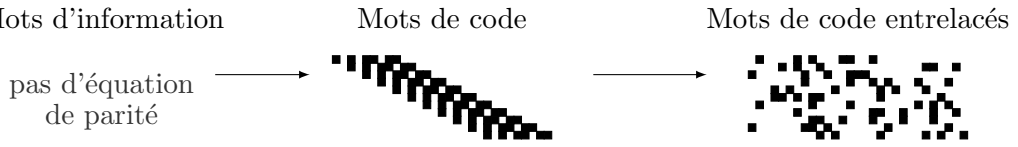


FIGURE 8.2 – Présence des équations de parité

— M mots d'information $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^M$ sont choisis aléatoirement, indépendamment les uns des autres et suivant une distribution uniforme.
— Les M mots de code entrelacés et bruités $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ sont donnés.

Ces hypothèses sont assez faibles. En effet, la seule valeur connue est la longueur N de l'entrelaceur. Cette longueur correspondant à la longueur des mots observés est généralement connue. Si ce n'est pas le cas, elle peut être retrouvée à l'aide des méthodes présentées dans [55, 73–75, 92]. Concernant le code convolutif, dont on ne connaît rien a priori, il peut être remarqué qu'il n'est pas forcément systématique et qu'il peut même être poinçonné à condition que le motif de poinçonnage soit périodique. De plus, il n'est pas nécessaire de connaître ce motif.

Utiliser un code convolutif puis un entrelaceur est courant, cet enchaînement est normé, par exemple dans les standards IEEE 802.11n [65], 802.16e [68], 802.22 [70], GMR-1 [47] ou encore WiMedia [141]. Dans toutes ces normes, comme dans les autres, les entrelaceurs utilisés sont structurés mais cette structure diffère. Le fait de supposer l'utilisation d'une permutation aléatoire permet de reconstruire l'entrelaceur quelle que soit sa structure.

Le code convolutif assure la présence de nombreuses équations de parité de petit poids. En effet, comme rappelé dans le chapitre 2, les équations d'un code convolutif de type (n, k) de mémoire faible ou modérée sont de petit poids et sont vérifiées tout au long du mot de code en les décalant de n en n. L'entrelaceur qui est appliqué sur les mots de code conserve la présence de ces équations mais pas la régularité entre elles. Schématiquement, la situation peut être représentée comme sur la figure 8.2. Sur cette figure, comme sur toutes les suivantes, chaque ligne représente une équation de parité et un petit carré en couleur indique une position intervenant dans cette équation.

Ce sont ces équations et leur régularité avant entrelacement qui sont utilisées pour reconstruire l'entrelaceur et retrouver le code convolutif.

8.2 Grandes lignes de l'algorithme

Notations. *À partir de maintenant, le code $\mathcal{C}$ entrelacé par π est noté $\mathcal{C}_\pi$. Une équation de parité $\mathcal{E}$ du code $\mathcal{C}_\pi$ est parfois vue comme un ensemble d'indices $\{e_1, \ldots, e_t\}$, ces indices sont tels que $m^s_{e_1} \oplus m^s_{e_2} \oplus \cdots \oplus m^s_{e_t} = 0$ pour tout $s \in \{1, \ldots, M\}$. Cette notation sous forme d'ensemble peut également être utilisée pour des équations du code $\mathcal{C}$. Dans ce cas, ils sont tels que $v^s_{e_1} \oplus v^s_{e_2} \oplus \cdots \oplus v^s_{e_t} = 0$ pour tout $s \in \{1, \ldots, M\}$.*
Avec cette notation, appliquer la permutation π sur une équation $\mathcal{E} = \{e_1, \ldots, e_t\}$ revient à déterminer $\pi(\mathcal{E}) = \{\pi(e_1), \ldots, \pi(e_t)\}$. Enfin, l'équation $\mathcal{E}'$ obtenue en décalant l'équation $\mathcal{E} = \{e_1, \ldots, e_t\}$ de in positions vers la droite (avec i un entier) est définie par :
$\mathcal{E}' = \{e_1 + in, e_2 + in, \ldots, e_t + in\}$.

L'algorithme proposé pour la reconstruction de codes convolutifs entrelacés se découpe en quatre grandes étapes :

1. **Rechercher un ensemble $\mathcal{L}$ d'équations de parité de $\mathcal{C}_\pi$ de poids t,** t étant le plus petit poids tel que le code $\mathcal{C}_\pi$ satisfait un nombre linéaire (en N) d'équations de parité de poids t.
2. **Classer les équations de $\mathcal{L}$ en groupes** $\mathcal{L}^1, \mathcal{L}^2, \ldots$ Chaque groupe contient des équations qui, une fois désentrelacées, sont obtenues par décalage d'un multiple de n les unes par rapport aux autres.
3. **Retrouver une équation de parité du code $\mathcal{C}$.** Soit $\mathcal{L}^1$ le groupe d'équations de parité qui a la plus petite taille de voisinage. La taille du voisinage d'une équation $\mathcal{E}$ est simplement le nombre d'équations de $\mathcal{L}$ qui ont au moins un indice en commun avec $\mathcal{E}$. Presque toutes les équations appartenant à un même groupe ont la même taille de voisinage, c'est cette taille de voisinage qui est associée au groupe correspondant. En utilisant seulement ce groupe $\mathcal{L}^1$ et des considérations de théorie des graphes, une équation de parité de $\mathcal{C}$ est alors retrouvée.
4. **Ordonner les équations de parité de $\mathcal{L}^1$.** Par construction, les ℓ équations de $\mathcal{L}^1$ correspondent après désentrelacement à ℓ équations de parité du code $\mathcal{C}$ qui sont les décalages les unes des autres par un multiple de n. Ordonner les équations revient à rechercher l'ordonnancement $\mathcal{A} = \mathcal{E}_{a_1}, \ldots, \mathcal{E}_{a_\ell}$ des équations de $\mathcal{L}^1$ tel que $\pi^{-1}(\mathcal{E}_{a_i})$ soit égale à $\pi^{-1}(\mathcal{E}_{a_1})$ décalée de $n(i-1)$ positions vers la droite.
5. **Reconstruire π.** Les équations étant ordonnées, il faut maintenant ordonner les positions afin de retrouver l'entrelaceur.

Dans cet algorithme, la première étape consiste à utiliser l'algorithme de Dumer modifié, présenté dans le chapitre 6. Le fait de choisir t tel que le code $\mathcal{C}_\pi$ satisfait un nombre linéaire (en N) d'équations de parité de poids t permet simplement de ne pas tenir compte des équations de parité impliquant les positions aux extrémités des mots de code (non entrelacés) et dues à une initialisation constante des registres du codeur. Prendre le plus petit poids t sert uniquement à réduire le coût de la recherche de cet ensemble d'équations. De plus, utiliser la méthode de Dumer modifiée permet de retrouver un ensemble d'équations de parité même si les données $\tilde{\boldsymbol{m}}^1, \ldots, \tilde{\boldsymbol{m}}^M$ sont bruitées. Cette phase de recherche d'équations de parité de petit poids étant exactement la même que précédemment, elle n'est pas détaillée ici contrairement aux autres étapes de l'algorithme.

8.3 Tri des équations de parité

Le but de cette étape est de classer les équations de $\mathcal{L}$ en différents groupes tels que les équations dans un même groupe soient du même type.

Définition 8.1 (Équations du même type dans $\mathcal{C}$)**.** *Deux équations de parité $\mathcal{E}$ et $\mathcal{E}'$ du code $\mathcal{C}$ sont dites du même type si l'une est le décalé de l'autre par un multiple de n. Cette relation est notée $\mathcal{E} \sim \mathcal{E}'$.*
Vu sous forme d'ensemble, si $\mathcal{E} = \{e_1, \ldots, e_t\}$ et $\mathcal{E}' = \{e'_1, \ldots, e'_t\}$ alors $\mathcal{E} \sim \mathcal{E}'$ si et seulement si il existe un entier i tel que $\{e'_1, \ldots, e'_t\} = \{e_1 + in, \ldots, e_t + in\}$.
Toutes les équations de parité d'un même type définissent une classe d'équivalence.

Par extension, il s'ensuit la définition suivante sur les types des équations de $\mathcal{C}_\pi$.

Définition 8.2 (Équations du même type dans $\mathcal{C}_\pi$)**.** *Deux équations de parité $\mathcal{E}$ et $\mathcal{E}'$ du code $\mathcal{C}_\pi$ sont dites du même type si et seulement si $\pi^{-1}(\mathcal{E}) \sim \pi^{-1}(\mathcal{E}')$.*

Pour retrouver la permutation et le code convolutif, la structure des matrices de parité des codes convolutifs est utilisée. Le but est donc de trouver un sous-ensemble d'équations de $\mathcal{L}$ telles qu'une fois réordonnées et désentrelacées la matrice obtenue corresponde à une seule et même ligne qui se décale de n en n. Pour cela, différents groupes sont formés, l'un d'entre eux est ensuite choisi et permet la reconnaissance de $\mathcal{C}$ et la reconstruction de π.

En fonction du code convolutif utilisé, les mots de code satisfont un nombre plus ou moins important d'équations de parité de même poids mais de type différent. Les deux exemples suivants permettent d'illustrer cette situation.

Exemple. *Le code convolutif utilisé dans la norme DVB-S2 est défini par $(1 + D + D^2 + D^3 + D^6, 1 + D^2 + D^3 + D^5 + D^6)$. Ce code satisfait 11 types d'équations de poids 10 et aucune équation de poids inférieur. La liste suivante donne une équation de chacune des classes d'équivalence.*

- $\mathcal{E}_1 = \{1, 2, 4, 7, 8, 9, 10, 11, 13, 14\}$
- $\mathcal{E}_2 = \{1, 2, 3, 6, 7, 8, 12, 14, 15, 16\}$
- $\mathcal{E}_3 = \{1, 2, 3, 5, 7, 11, 13, 16, 17, 18\}$
- $\mathcal{E}_4 = \{1, 2, 3, 6, 10, 12, 13, 17, 19, 20\}$
- $\mathcal{E}_5 = \{1, 2, 4, 9, 11, 15, 16, 17, 19, 20\}$
- $\mathcal{E}_6 = \{1, 2, 4, 5, 6, 7, 16, 19, 21, 22\}$
- $\mathcal{E}_7 = \{1, 2, 4, 10, 11, 12, 18, 20, 21, 22\}$
- $\mathcal{E}_8 = \{1, 2, 4, 10, 14, 17, 19, 22, 23, 24\}$
- $\mathcal{E}_9 = \{1, 2, 3, 5, 7, 12, 22, 24, 25, 26\}$
- $\mathcal{E}_{10} = \{1, 2, 4, 9, 12, 14, 22, 25, 27, 28\}$
- $\mathcal{E}_{11} = \{1, 2, 4, 10, 14, 18, 28, 30, 31, 32\}$

Il peut être remarqué qu'elle sont toutes obtenues en additionnant la première équation de cette liste avec un certain nombre de ses décalés (par un multiple de 2). Pour cet exemple, 11 groupes d'équations de parité sont formés lors du tri.

Exemple. *Le code convolutif défini par $(1+D+D^2+D^5, 1+D+D^3+D^4+D^6)$ satisfait un seul type d'équations de parité de poids 8 et aucune équation de parité de poids plus faible. Lors de la recherche des équations de parité de petits poids, seules les équations de ce type sont retrouvées, l'étape de tri n'est alors pas nécessaire. Néanmoins, comme le nombre de types d'équations satisfaites par le code $\mathcal{C}$ n'est pas connu a priori, le tri est donc tout de même appliqué et un seul groupe d'équations est formé.*

En utilisant la définition suivante, il est assez simple de trier les équations de parité du code $\mathcal{C}$.

Définition 8.3 (Longueur d'une équation)**.** *Soit $\mathcal{E} = \{e_1, \ldots, e_t\}$ une équation de parité, sa longueur s est définie par :*

$$s = \max_{1 \leq i \leq t}(e_i) - \min_{1 \leq i \leq t}(e_i) + 1$$

Pour un code convolutif non entrelacé, toutes les équations du même type possèdent la même longueur. Mais cette propriété est perdue lorsque le code est entrelacé, d'où la nécessité de trouver un nouveau moyen d'effectuer ce tri.

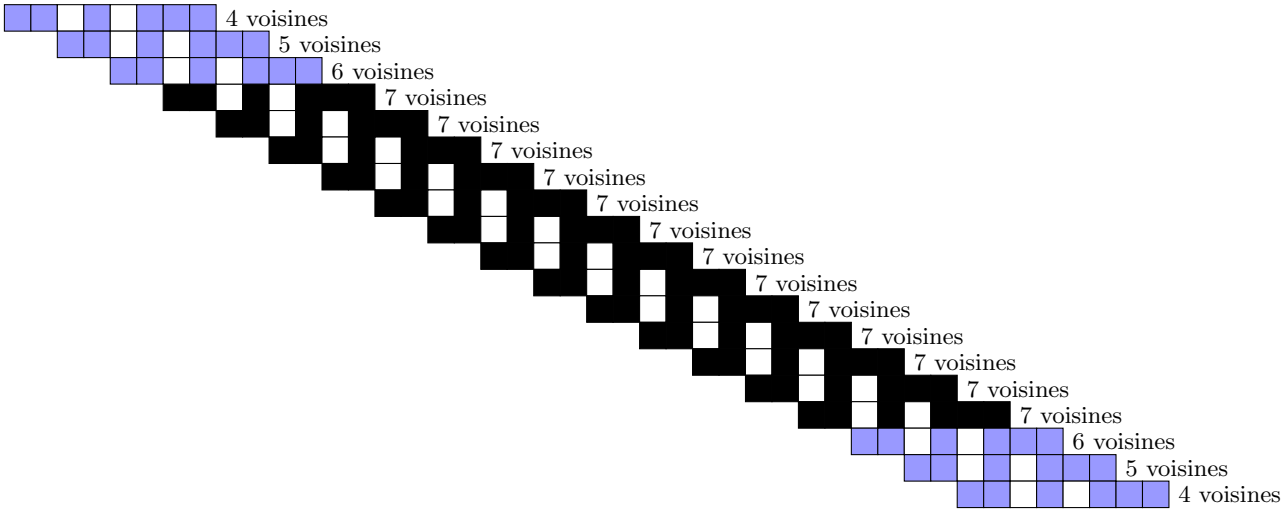


FIGURE 8.3 – Les équations aux extrémités ont des tailles de voisinage plus faibles

8.3.1 Tri par histogramme

Une des variables restant invariante avant et après entrelacement est la taille des intersections entres les différentes équations. Une idée pour classer les équations de parité est donc d'utiliser ce critère.

Définition 8.4. *(Voisinage direct) L'équation $\mathcal{E}'$ est dans le voisinage direct de l'équation $\mathcal{E}$ si et seulement si ces deux équations ont au moins un indice en commun.*

Définition 8.5. *(Taille du voisinage) La taille du voisinage de l'équation $\mathcal{E}$ est le nombre d'équations dans son voisinage direct.*

À l'exception des équations faisant intervenir des positions placées aux extrémités (avant entrelacement), toutes les équations du même type ont la même taille de voisinage et cette propriété est vérifiée même après l'entrelacement.

Concernant les équations faisant intervenir des positions placées aux extrémités, elles ont une taille de voisinage moins importante que les autres équations du même type, comme illustré sur la figure 8.3. Sur cette figure, un seul type d'équation est représenté. Le principe est le même lorsque plusieurs types équations sont présents.

Une méthode de tri est alors de classer les équations en fonction de la taille de leur voisinage. Pour cela, l'histogramme représentant pour chaque taille de voisinage le nombre d'équations concernées est tracé. Sur cet histogramme, différents pics sont présents, ils représentent les différents types d'équations. De plus, quelques équations ont une taille de voisinage ne correspondant à aucun des pics : il s'agit des équations faisant intervenir des bits placés aux extrémités (avant entrelacement). Ces équations ne peuvent pas être associées à l'un des ensembles formés, et ne sont donc pas classées.

Exemple. *Le code convolutif défini par $(1 + D^3, 1 + D + D^2 + D^3)$ satisfait deux types d'équation de parité de poids 6. Si $N = 42$ alors l'histogramme obtenu est celui de la figure*

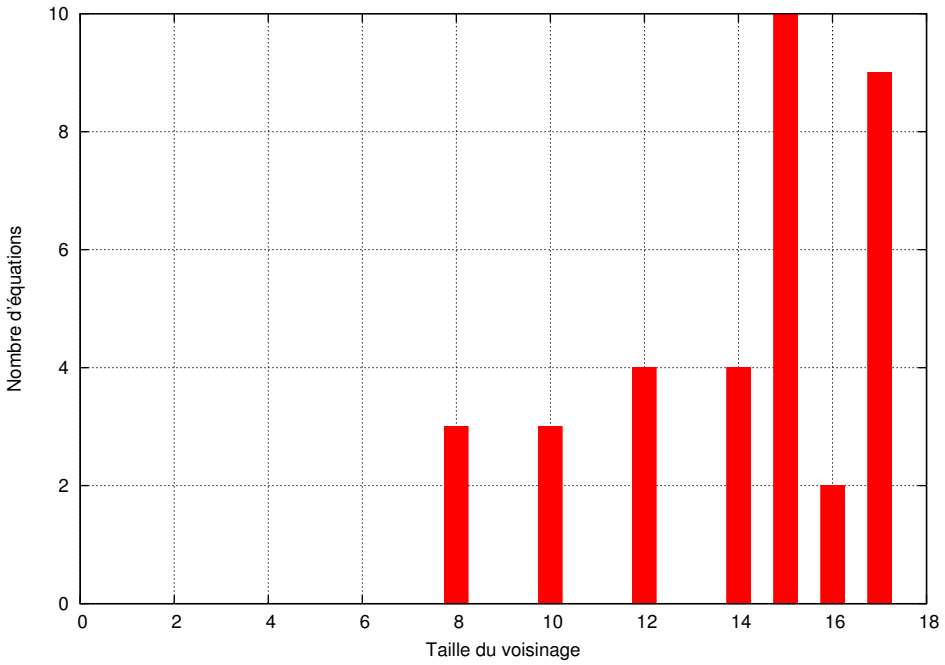


FIGURE 8.4 – Histogramme des tailles de voisinage pour le code $(1 + D^3, 1 + D + D^2 + D^3)$ et $N = 42$

8.4. Deux pics sont visibles et correspondent aux deux types d'équations de parité. Pour cet exemple, 16 équations de parité sont "non-classées". Remarquons que si l'entrelaceur est plus long, les pics sont plus importants mais le nombre d'équations non-classées reste constant.

Cette méthode de tri est très simple et nécessite seulement de déterminer la taille du voisinage de chaque équation. Cependant, elle présente un inconvénient important : dans les ensembles d'équations formés, il peut se trouver des équations d'autres types faisant intervenir des bits placés aux extrémités. Les ensembles obtenus sont donc composés majoritairement d'équations de parité du même type mais des équations d'autres types peuvent être présentes et des équations du type majoritaire sont manquantes. L'existence de ces équations incorrectement classées explique la difficulté à former les groupes souhaités.

8.3.2 Tri par profil de voisinage

Afin d'éviter ces erreurs de classement, nous allons donner un critère de tri plus précis. L'idée est d'utiliser le profil de voisinage de chaque équation.

Définition 8.6 (Profil de voisinage)**.** *Le profil de voisinage $\mathcal{P}^{\mathcal{E}}$ d'une équation $\mathcal{E} \in \mathcal{L}$ est un vecteur de taille t : $\mathcal{P}^{\mathcal{E}} = (\mathcal{P}_1^{\mathcal{E}}, \ldots, \mathcal{P}_t^{\mathcal{E}})$ tel que $\mathcal{P}_i^{\mathcal{E}} = \#\{\mathcal{E}' \in \mathcal{L}$ et $\#\{\mathcal{E} \cap \mathcal{E}'\} = i\}$.*

En d'autres termes, ce profil encode la distribution des intersections avec une équation $\mathcal{E}$ donnée. Il est beaucoup plus précis que la taille du voisinage. Toutes les équations du même type (hormis celles faisant intervenir des bits placés aux extrémités) ont le même profil. De plus, la taille du voisinage se déduit du profil puisque, par construction, elle est égale à la somme des termes du vecteur définissant le profil de voisinage.

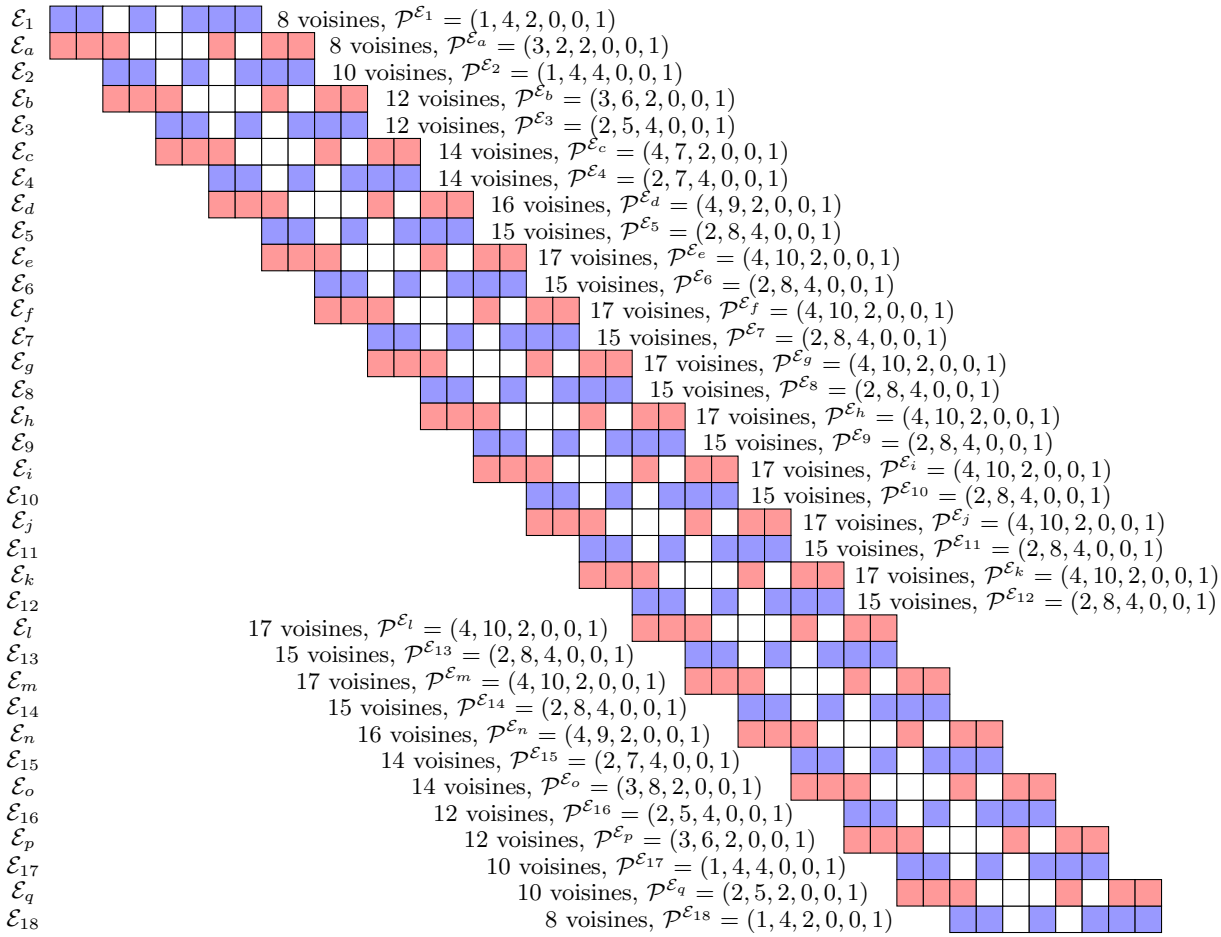


FIGURE 8.5 – Profils de voisinage des équations du code défini par $(1 + D^3, 1 + D + D^2 + D^3)$, pour une longueur $N = 42$.

Exemple. *Avec le même code que pour l'exemple précédent et toujours une longueur $N = 42$, les profils de chaque équation sont reportés sur la figure 8.5. Les deux couleurs représentent les deux types d'équation de parité. Tous les profils de voisinage finissent par un 1 puisqu'une équation est considérée être dans son propre voisinage. Évidemment, ces profils de voisinage sont les mêmes lorsque le code est entrelacé, puisque l'entrelacement correspond à une simple permutation des colonnes de ce schéma.*

Avec ce profil de voisinage, les équations faisant intervenir des bits placés aux extrémités peuvent également être classées (au moins en partie). En effet, ces équations ont un profil "inférieur" à celui des autres équations au sens de la relation d'ordre suivante :

Définition 8.7 (Ordre partiel sur les profils de voisinage)**.** *L'ordre partiel entre deux profils de voisinage $\mathcal{P}$ et $\mathcal{P}'$ est défini par :*

$$\mathcal{P} \leq \mathcal{P}' \text{ si et seulement si } \mathcal{P}_i \leq \mathcal{P}'_i \quad \forall i \in \{1, \ldots, t\}$$

Cela nous conduit à une nouvelle définition d'un type. Nous dirons que $\mathcal{E}$ et $\mathcal{E}'$ sont du même type si et seulement si $\mathcal{P}^{\mathcal{E}} \leq \mathcal{P}^{\mathcal{E}'}$ ou $\mathcal{P}^{\mathcal{E}'} \leq \mathcal{P}^{\mathcal{E}}$. On en déduit l'algorithme 8 pour classer

Algorithme 8 : TRI_ÉQUATIONS_DE_PARITÉ

Entrée :
$\mathcal{L}$ la liste des équations de parité de poids t satisfaites par $\mathcal{C}_\pi$
Sortie :
$\mathcal{L}^1, \mathcal{L}^2, \ldots$ les groupes d'équations de parité triées
$\mathcal{L}^{NC}$ la liste des équations de parité non-classées $//\mathcal{L}^{NC} = \mathcal{L} \setminus \{\mathcal{L}^1, \mathcal{L}^2, \ldots\}$

Pour $\mathcal{E} \in \mathcal{L}$
$\mathcal{P}^{\mathcal{E}} \leftarrow$ le profil de voisinage de $\mathcal{E}$
$\mathcal{P}^{T_1}, \mathcal{P}^{T_2}, \ldots, \mathcal{P}^{T_r} \leftarrow$ les profils de voisinage types
//c'est-à-dire les profils de voisinage significativement plus fréquents
$\mathcal{L}^1, \mathcal{L}^2, \ldots, \mathcal{L}^r \leftarrow \emptyset$
Pour $\mathcal{E} \in \mathcal{L}$
Si il existe un unique $i \in \{1, \ldots, r\}$ tel que $\mathcal{P}^{\mathcal{E}} \leq \mathcal{P}^{T_i}$
$\mathcal{L}^i \leftarrow \mathcal{L}^i \cup \{\mathcal{E}\}$
Sinon
$\mathcal{L}^{NC} \leftarrow \mathcal{L}^{NC} \cup \{\mathcal{E}\}$
Retourner $\mathcal{L}^1, \ldots, \mathcal{L}^r, \mathcal{L}^{NC}$

les équations de parité.

Le nombre, noté r, de groupes d'équations de parité formés par l'algorithme n'est pas connu a priori. Sa valeur se déduit de la liste des profils de voisinage. En effet, dans cette liste certains profils sont, par construction, beaucoup plus représentés que les autres mais leur nombre varie selon le code $\mathcal{C}$ utilisé. Les seules équations qui n'ont pas un de ces profils, dit profils types, sont des équations faisant intervenir des indices placés aux extrémités (avant entrelacement). La condition sur l'unicité de i tel que $\mathcal{P}^{\mathcal{E}} \leq \mathcal{P}^{T_i}$ permet tout de même de classer ces équations mais surtout de ne pas faire d'erreur de classement.

Puisque quelques équations de parité ne peuvent pas être affectées de façon certaine à l'un des groupes, il arrive que des groupes ne soient pas complets. C'est pour cette raison que les équations "non-classées" sont conservées, elle serviront en fin de reconstruction de l'entrelaceur. À ce moment là, des informations supplémentaires sur le code et l'entrelaceur seront connues, il sera alors plus facile de classer ces équations résiduelles.

Exemple. *Sur l'exemple de la figure 8.5, deux profils sont beaucoup plus fréquents que les autres :* $\mathcal{P}^{T_1} = (2, 8, 4, 0, 0, 1)$ *et* $\mathcal{P}^{T_2} = (4, 10, 2, 0, 0, 1)$*. Deux groupes d'équations sont alors formés. Le premier contient les équations* $\{\mathcal{E}_2, \mathcal{E}_3, \ldots, \mathcal{E}_{16}, \mathcal{E}_{17}\}$ *et le second les équations* $\{\mathcal{E}_a, \mathcal{E}_b, \ldots, \mathcal{E}_o, \mathcal{E}_p\}$*. Finalement,* $\mathcal{L}^{NC}$ *contient seulement trois équations* $\{\mathcal{E}_1, \mathcal{E}_q, \mathcal{E}_{18}\}$ *et aucune erreur de tri n'est commise. À titre de comparaison, par la méthode basée sur l'histogramme, nous avions* 16 *équations non-classées.*

Remarque. *Si* $\mathcal{L}$ *ne contient pas toutes les équations de parité de poids* t *satisfaites par* $\mathcal{C}_\pi$ *alors cette méthode de tri reste efficace. Effectivement, les équations voisines des équations*

manquantes sont classées de la même façon que les équations faisant intervenir des positions aux extrémités, leur profil vérifiant la relation d'ordre partiel avec le profil des équations du même type.

Remarque. *Lorsque les mémoires du codeur convolutif sont toujours initialisées de la même façon (par exemple fixées à* 0*), l'ensemble* $\mathcal{L}$ *contient des équations de poids* t *qui sont uniquement dues à l'initialisation. Afin de permettre la reconstruction de l'entrelaceur, ces équations ne doivent pas être classées dans l'un des groupes formés. En appliquant la méthode de tri précédente, ces équations "parasites" sont placées dans* $\mathcal{L}^{NC}$ *mais il y a un nombre plus important d'équations non-parasites dans* $\mathcal{L}^{NC}$*. C'est en fin de reconstruction que ces équations seront triées et utilisées.*

8.3.3 Choix d'un ensemble d'équations

Pour la suite de la reconnaissance du code convolutif entrelacé un seul des groupes d'équations formés est utilisé. Le choix de ce groupe peut être fait de la manière suivante, cependant, si ce choix est effectué d'une autre façon, la suite de l'algorithme de reconstruction n'en est pas altérée.

Une des étapes de l'algorithme consiste à tester toutes les équations de poids t et de longueur inférieure à un seuil s_{max} où s_{max} est la longueur maximale de l'équation de parité du code $\mathcal{C}$. C'est pour cette raison qu'il est intéressant de choisir le groupe d'équations $\mathcal{L}^i$ tel qu'une fois désentrelacées, ses équations soient de longueur minimale. Il n'est pas possible de déterminer de façon certaine quel est ce groupe, mais il est facile de choisir le groupe qui est probablement celui minimisant la longueur des équations après désentrelacement. En effet, plus une équation est longue, plus la taille de son voisinage risque d'être important, d'où le choix du groupe $\mathcal{L}^i$ tel que la taille du voisinage des équations est minimale. Autrement dit, le groupe d'équations considéré pour la suite de la reconstruction est le groupe $\mathcal{L}^i$ tel que

$$\sum_{1\leq j\leq t} \mathcal{P}_j^{T_i} \leq \sum_{1\leq j\leq t} \mathcal{P}_j^{T_k} \quad \forall k \in \{1,\ldots,r\}$$

Ce groupe est noté $\mathcal{L}^1$ dans la suite de ce chapitre. De plus, il correspond généralement au groupe contenant le plus d'équations de parité. En effet, les risques de ne pas classer une équation de petite longueur sont plus faibles que pour une équation de longueur plus importante.

8.3.4 Déduction de la longueur n du code convolutif

Le nombre d'équations dans $\mathcal{L}^1$ permet de déterminer la valeur de n. Effectivement, $\mathcal{L}^1$ contient toutes (ou presque) les équations d'une même classe d'équivalence. Après désentrelacement, ces équations se décalent de n en n. D'où :

$$n = \lfloor \frac{N}{\#\mathcal{L}^1} \rfloor$$

avec $\#\mathcal{L}^1$ représentant le nombre d'équations de parité contenues dans $\mathcal{L}^1$ et N la longueur de l'entrelaceur.

8.4 Identification du code convolutif

Hypothèses. Désormais, un ensemble $\mathcal{L}^1$ de ℓ équations de parité de $\mathcal{C}_\pi$ appartenant à la même classe d'équivalence est connu. Une fois désentrelacées, ces équations correspondent à ℓ équations consécutives satisfaites par le code $\mathcal{C}$ (remarquons que pour qu'elles soient consécutives aucune équation intermédiaire n'est manquante). Pour la suite de cette section, ces hypothèses sont regroupées sous le nom d'hypothèse $\boldsymbol{\mathcal{H}_1}$.

Notations. *Les équations de $\mathcal{L}^1$ sont utilisées afin de retrouver une équation de parité $\mathcal{E}_\mathcal{C}$ de $\mathcal{C}$ telle que $\mathcal{E}_\mathcal{C} \sim \pi^{-1}(\mathcal{E})$ pour toutes les équations $\mathcal{E} \in \mathcal{L}^1$. L'équation de parité $\mathcal{E}_\mathcal{C}$ définit un code convolutif $\mathcal{D}$ de longueur n et de dimension $n-1$. La longueur maximale de $\mathcal{E}_\mathcal{C}$ est notée s_{max}.*

Le but est de retrouver le code $\mathcal{D}$. Pour cela, tous les codes convolutifs $(n, n-1)$ possédant une équation de parité de poids t et de longueur inférieure à s_{max} sont testés. Ce test repose sur l'association d'un graphe à un ensemble d'équations de parité donné. Ce graphe est tel que :

— le graphe associé à la classe d'équivalence d'une équation de parité $\mathcal{E}$ d'un code convolutif $(n, n-1)$ discrimine le code convolutif.
— si deux ensembles d'équations diffèrent par une permutation sur leurs indices alors leurs graphes sont équivalents.

Remarque. *La notion d'équivalence de graphes est définie plus loin (dans la sous-section 8.4.1).*

Le code $\mathcal{D}$ est retrouvé en recherchant une équivalence entre le graphe associé à $\mathcal{L}^1$ et le graphe associé à chaque code convolutif $(n, n-1)$.

Si le code convolutif $(n, n-1)$ testé est défini par l'équation de parité $\mathcal{E} = \{e_1, \ldots, e_t\}$ alors l'ensemble

$$\mathcal{L}^\mathcal{E} = \{\{e_1 + in, e_2 + in, \ldots, e_t + in\}, 0 \leq i < \ell\}$$

contient ℓ équations de parité de ce code qui sont les décalées les unes des autres. C'est à cet ensemble qu'un graphe est associé. Par abus il est aussi appelé graphe associé à $\mathcal{E}$.

Notation. *L'équation obtenue en décalant l'équation $\mathcal{E}$ par in est notée $\mathcal{E}_{(i)}$. Avec cette notation, l'ensemble d'équations associé à $\mathcal{E}$ est défini par : $\mathcal{L}^\mathcal{E} = \{\mathcal{E}_{(0)}, \mathcal{E}_{(1)}, \ldots, \mathcal{E}_{(\ell-1)}\}$.*

Remarque. *L'équation $\mathcal{E}$ est définie par $\{e_1, \ldots, e_t\}$. Supposons que $e_1 < e_2 < \cdots < e_t$. Le but étant de retrouver une équation de parité définissant le code $\mathcal{D}$, seules les équations telles que $e_1 \leq n$ sont testées. En effet, si une telle équation est une équation de parité du code $\mathcal{D}$, alors nécessairement les équations obtenues en la décalant d'un multiple de n sont également des équations de parité de $\mathcal{D}$ et réciproquement. Il est donc inutile de tester les équations obtenues en décalant une équation déjà testée par un multiple de n, d'où la restriction aux équations telles que $e_1 \leq n$.*

Il s'en déduit l'algorithme 9 de recherche de l'équation $\mathcal{E}_\mathcal{C}$. L'algorithme retourne un ensemble d'équations compatibles. En pratique, généralement, $2 \times n!$ équations sont retournées.

Les codes engendrés par ces équations sont équivalents et indistinguables. En effet, ces $2 \times n!$ équations correspondent à l'équation du code recherché ainsi qu'aux équations obtenues en la lisant en sens inverse et/ou à l'équation de parité du code généré en permutant les n sorties du codeur convolutif.

Algorithme 9 : IDENTIFICATION_DU_CODE_CONVOLUTIF

Entrée :
$\mathcal{L}^1$ la liste des équations de parité de poids t d'une même classe d'équivalence
Sortie :
A une liste d'équations compatibles (avec l'assurance que $\mathcal{E}_\mathcal{C} \in A$)

$A \leftarrow \emptyset$
$\tilde{\mathcal{G}}(\mathcal{L}^1) \leftarrow$ le graphe associé à $\mathcal{L}^1$
Pour toutes les équations $\mathcal{E}$ de poids t et de longueur inférieure à s_{max}
  $\tilde{\mathcal{G}}(\mathcal{L}^\mathcal{E}) \leftarrow$ le graphe associé à $\mathcal{E}$
  Si $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^\mathcal{E})$ sont équivalents
    $A \leftarrow A \cup \{\mathcal{E}\}$
Retourner A

Graphes associés à un ensemble d'équations

Définition 8.8 (Graphe associé à un ensemble d'équations)**.** *Le graphe $\tilde{\mathcal{G}}(\mathcal{L})$ associé à l'ensemble d'équations $\mathcal{L}$ est tel que :*

- *Chaque équation de $\mathcal{L}$ est représentée par un sommet.*
- *Si deux équations $\mathcal{E}$ et $\mathcal{E}'$ de $\mathcal{L}$ ont k indices en commun (autrement dit $\#\{\mathcal{E} \cap \mathcal{E}'\} = k$) alors les sommets représentant $\mathcal{E}$ et $\mathcal{E}'$ sont reliés par k arêtes.*
- *Chaque arête est étiquetée par l'indice qu'elle représente.*

Notation. *Le graphe $\tilde{\mathcal{G}}(\mathcal{L})$ sans étiquettes sur ses arêtes est noté $\mathcal{G}(\mathcal{L})$.*

Exemple. *Soit l'ensemble $\mathcal{L}$ contenant les quatre équations $\mathcal{E}^1 = \{1, 4, 6\}$, $\mathcal{E}^2 = \{2, 4, 5\}$, $\mathcal{E}^3 = \{4, 6, 7\}$ et $\mathcal{E}^4 = \{2, 5, 7\}$. Le graphe $\tilde{\mathcal{G}}(\mathcal{L})$ associé à $\mathcal{L}$ est représenté sur la figure 8.6.*

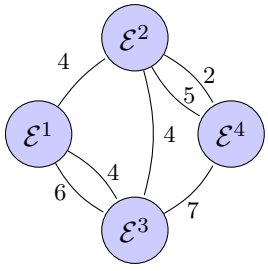


FIGURE 8.6 – Graphe associé à l'ensemble d'équations :
$\{\mathcal{E}^1 = \{1, 4, 6\}, \mathcal{E}^2 = \{2, 4, 5\}, \mathcal{E}^3 = \{4, 6, 7\}, \mathcal{E}^4 = \{2, 5, 7\}\}$

Notation. *Le graphe associé à $\mathcal{L}^1$ est noté $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et celui associé à l'ensemble d'équations $\mathcal{L}^{\mathcal{E}}$ déduit de l'équation testée $\mathcal{E}$ est noté $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$.*

8.4.1 Comparaison de graphes

Afin de comparer les graphes ainsi construits, les deux définitions suivantes sont nécessaires.

Définition 8.9 (Graphes isomorphes)**.** *Deux graphes $\mathcal{G}$ et $\mathcal{G}'$ sont isomorphes si et seulement si il existe un isomorphisme ϕ entre les sommets de $\mathcal{G}$ et les sommets de $\mathcal{G}'$ tel que pour tous les couples (x, y) de sommets dans $\mathcal{G}$ il y a autant d'arêtes entre x et y qu'entre $\phi(x)$ et $\phi(y)$ dans $\mathcal{G}'$. Autrement dit, deux graphes sont isomorphes s'ils sont égaux à une numérotation de sommet près.*

Cette définition s'étend aux graphes étiquetés :

Définition 8.10 (Graphes équivalents)**.** *Deux graphes $\tilde{\mathcal{G}}$ et $\tilde{\mathcal{G}}'$ sont équivalents si et seulement si les deux conditions suivantes sont satisfaites*

- *$\tilde{\mathcal{G}}$ et $\tilde{\mathcal{G}}'$ sont isomorphes (l'isomorphisme est noté ϕ).*
- *Il existe une bijection ψ entre les étiquettes de $\tilde{\mathcal{G}}$ et celles de $\tilde{\mathcal{G}}'$ telle que pour tous les couples (x, y) de sommets de $\tilde{\mathcal{G}}$, si les étiquettes des arêtes entre x et y sont notées $\{a_1, \ldots, a_s\}$ alors les arêtes entre $\phi(x)$ et $\phi(y)$ sont étiquetées par $\{\psi(a_1), \ldots, \psi(a_s)\}$.*

Pour retrouver l'équation de parité $\mathcal{E}_{\mathcal{C}}$ du code $\mathcal{D}$ la proposition suivante est utilisée :

Proposition 8.11. *Sous l'hypothèse $\boldsymbol{\mathcal{H}}_1$, les graphes $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ sont équivalents.*

Démonstration. L'hypothèse $\boldsymbol{\mathcal{H}}_1$ implique que $\mathcal{L}^1$ contient les mêmes équations de parité que $\mathcal{L}^{\mathcal{E}_{\mathcal{C}}}$ mais entrelacées par la permutation $\pi : \mathcal{L}^1 = \{\pi(\mathcal{E}_{\mathcal{C}(0)}), \pi(\mathcal{E}_{\mathcal{C}(1)}), \ldots, \pi(\mathcal{E}_{\mathcal{C}(\ell-1)})\}$. Il s'en déduit deux isomorphismes entre ces graphes et donc leur équivalence : π donne directement ces deux isomorphismes.

$$\begin{array}{rccc} \phi : & \tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}}) & \to & \tilde{\mathcal{G}}(\mathcal{L}^1) \\ & \mathcal{E}_{\mathcal{C}(i)} & \mapsto & \pi(\mathcal{E}_{\mathcal{C}(i)}) \end{array} \quad \text{et} \quad \begin{array}{rccc} \psi : & \tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}}) & \to & \tilde{\mathcal{G}}(\mathcal{L}^1) \\ & i & \mapsto & \pi(i) \end{array}$$

□

Afin de mieux visualiser la situation, l'exemple suivant illustre les conséquences sur des graphes des différentes transformations possibles sur les équations.

Exemple. *Le code convolutif défini par $(1 + D^2, 1 + D + D^2)$ satisfait une équation de poids 5. Si quatre équations consécutives sont connues alors l'illustration de ces équations et le graphe qui leur est associé sont représentés sur la figure 8.7.*
Les équations retournées par l'algorithme de Dumer modifié ne sont pas nécessairement bien ordonnées. Dans ce cas les équations sont numérotées différemment, mais le graphe obtenu est équivalent à celui d'origine. La figure 8.8 représente cette situation, un isomorphisme entre les sommets des graphes est défini par $\phi(\mathcal{E}^1) = \mathcal{E}'^3$, $\phi(\mathcal{E}^2) = \mathcal{E}'^1$, $\phi(\mathcal{E}^3) = \mathcal{E}'^4$ et $\phi(\mathcal{E}^4) = \mathcal{E}'^2$, l'isomorphisme sur les étiquettes est l'identité.
Le code peut également être entrelacé, le graphe obtenu est équivalent aux deux graphes précédents. Sur la figure 8.9 un entrelaceur est appliqué, les colonnes sont donc simplement permutées. L'isomorphisme sur les sommets entre ce dernier graphe et le graphe d'origine est le même que précédemment, l'isomorphisme ψ suivant permet de montrer que ces graphes sont équivalents : $\psi(4) = 1'$, $\psi(5) = 6'$, $\psi(6) = 9'$, $\psi(7) = 4'$, $\psi(8) = 12'$ et $\psi(10) = 5'$.

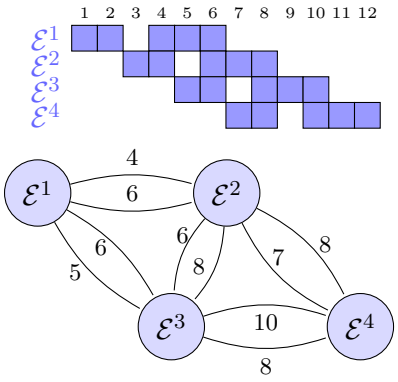


FIGURE 8.7 – 4 équations ordonnées.

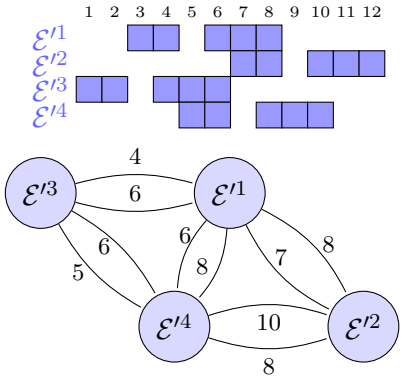


FIGURE 8.8 – 4 équations désordonnées.

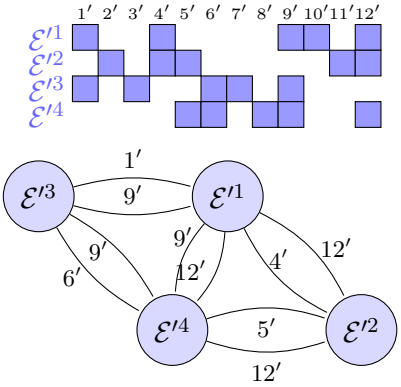


FIGURE 8.9 – Code entrelacé, 4 équations désordonnées.

8.4.2 Utilisation de sous-graphes

De par la régularité des codes convolutifs, si les listes des équations de parité étaient infinies les graphes obtenus seraient sommets-transitifs. C'est-à-dire que n'importe quel sommet peut être transformé en n'importe quel autre sommet par un isomorphisme. Les graphes utilisés ne sont pas infinis, mais sont presque sommets-transitifs au sens où seuls les sommets représentants des équations faisant intervenir des positions placées aux extrémités (avant entrelacement) ne peuvent pas être inversés avec d'autres sommets par un automorphisme. Cette régularité est tout de même utilisée en considérant des sous-graphes. Ceci permet d'effectuer les comparaisons beaucoup plus efficacement et de lever, en partie, l'hypothèse $\boldsymbol{\mathcal{H}}_1$.

L'objectif est donc d'utiliser de petits graphes, mais tout de même suffisamment discriminants. Ces sous-graphes sont des sous-graphes induits de $\mathcal{G}(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}(\mathcal{L}^1)$.

Définition 8.12 (Sous-graphe induit)**.** *Soit $\tilde{\mathcal{G}}$ un graphe et S un sous-ensemble de ses sommets. Le sous-graphe $\tilde{\mathcal{G}}'$ induit par S contient tous les sommets de S et toutes les arêtes de $\tilde{\mathcal{G}}$ qui possèdent leurs deux extrémités dans S.*

Définition 8.13 (Construction des sous-graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}^1(\mathcal{L}^1)$)**.** *Les sous-graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}^1(\mathcal{L}^1)$ sont les graphes induits par le voisinage à distance 1 d'un sommet de degré maximal respectivement dans $\mathcal{G}(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}(\mathcal{L}^1)$.*

Par construction, les sous-graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}^1(\mathcal{L}^1)$ sont tels que :

Définition 8.14 (Degré maximal d_1)**.** *d_1 est le degré maximal des sommets dans le graphe de taille infinie associé à l'équation $\mathcal{E}_{\mathcal{C}}$.*

Proposition 8.15. *Sous l'hypothèse $\boldsymbol{\mathcal{H}}_1$ et l'hypothèse que ℓ est suffisamment grand pour que le graphe $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ contienne un sommet de degré d_1, les graphes $\mathcal{G}^1(\mathcal{L}^1)$ et $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ sont isomorphes.*

Démonstration. Les hypothèses impliquent que $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ et $\mathcal{G}^1(\mathcal{L}^1)$ possèdent le même nombre de sommets, noté m. La construction de $\mathcal{G}^1(\mathcal{L}^1)$ entraîne l'existence d'entiers $j_1, j_2, \ldots, j_m$ tel que ce graphe contient des sommets représentants les équations $\{\pi(\mathcal{E}_{\mathcal{C}(j_1)}), \pi(\mathcal{E}_{\mathcal{C}(j_2)}), \ldots, \pi(\mathcal{E}_{\mathcal{C}(j_m)})\}$. Quant au graphe $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$, il existe un entier s tel que ses sommets représentent les équations $\{\mathcal{E}_{\mathcal{C}(j_1+s)}, \mathcal{E}_{\mathcal{C}(j_2+s)}, \ldots, \mathcal{E}_{\mathcal{C}(j_m+s)}\}$. D'où l'isomorphisme entre les deux graphes défini par : $\phi : \mathcal{G}^1(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}}) \to \mathcal{G}^1(\mathcal{L}^1) : \mathcal{E}_{\mathcal{C}(j_i+s)} \mapsto \pi(\mathcal{E}_{\mathcal{C}(j_i)})$ pour tout $i \in \{1, \ldots, m\}$. □

La recherche d'isomorphisme entre les sommets des graphes $\mathcal{G}^1(\mathcal{L}^1)$ et $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ se fait efficacement puisque ces graphes n'ont que peu de sommets :

Proposition 8.16. *Soient $\mathcal{E}$ une équation de parité d'un code convolutif $(n, n-1)$ et s la longueur de $\mathcal{E}$. Le sous-graphe $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ contient au maximum $2\lceil \frac{s}{n} \rceil - 1$ sommets.*

Remarque. *Une autre formulation de cette proposition est qu'une équation de parité $\mathcal{E}$, de longueur s, a au maximum $2\lceil \frac{s}{n} \rceil - 1$ équations dans son voisinage direct.*

Démonstration. Supposons que l'équation de degré maximal choisie pour construire $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ soit $\mathcal{E}_{(j)} = \{e_1, \ldots, e_t\}$ avec $e_1 < e_2 < \cdots < e_t$. s est la longueur de $\mathcal{E}$, d'où $s = e_t - e_1 + 1$. L'équation $\mathcal{E}_{(i)}$ est représentée dans $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ si et seulement si $\mathcal{E}_{(j)}$ et $\mathcal{E}_{(i)}$ ont au moins un indice en commun, c'est-à-dire si $\{e_1, \ldots, e_t\} \cap \{e_1 + in, \ldots, e_t + in\} \neq \emptyset$.

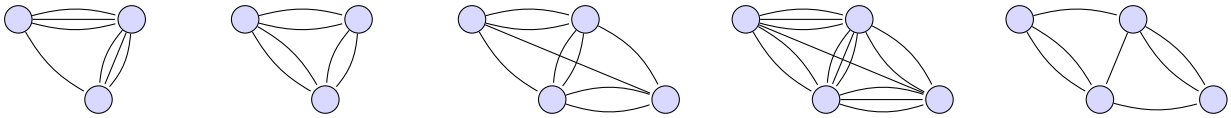


FIGURE 8.10 – Pour $n = 2$, les sous-graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ respectivement associés à $\mathcal{E} = \{1, 2, 3, 4, 5\}$, $\mathcal{E} = \{1, 2, 3, 5, 6\}$, $\mathcal{E} = \{1, 2, 4, 6, 7\}$, $\mathcal{E} = \{1, 2, 3, 5, 7\}$ et $\mathcal{E} = \{1, 2, 3, 6, 7\}$

Pour $i \geq 0$, si cette intersection est non vide alors $e_t \geq e_1 + in$. Il faut donc $0 \leq in < s$. Ceci implique que pour $i \geq 0$ il y a au maximum $\lceil \frac{s}{n} \rceil$ équations qui ont un indice en commun avec $\mathcal{E}_{(j)}$.
Pour $i < 0$, si l'intersection est non vide alors $e_t + in \geq e_1$ et cette inégalité est équivalente à $-s < in$. Pour $i < 0$, il y a donc au maximum $\lceil \frac{s}{n} \rceil - 1$ équations qui ont une position en commun avec $\mathcal{E}_{(j)}$.
Il s'en déduit la taille maximale du voisinage direct de $\mathcal{E}_{(j)}$: $\lceil \frac{s}{n} \rceil + \lceil \frac{s}{n} \rceil - 1$. □

Exemple. *Sur la figure 8.10 sont représentés quelques sous-graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ associés à des équations $\mathcal{E}$ de poids 5 lorsque $n = 2$.*

Ces graphes sont donc tout petit, ils sont tout de même très discriminants mais pas tout à fait suffisamment. Deux autres sous-graphes sont utilisés pour discriminer les derniers candidats potentiels.

Définition 8.17 (Construction des sous-graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$)**.** *Les sous-graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$ sont les graphes induits par le voisinage à distance $\leq$ 2 d'un sommet dont la taille de son voisinage à distance $\leq$ 2 est maximale, respectivement dans $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$ et $\tilde{\mathcal{G}}(\mathcal{L}^1)$.*

Ces graphes vérifient la proposition suivante :

Définition 8.18 (Voisinage à distance 2 maximal d_2)**.** *d_2 est la taille maximale du voisinage à distance $\leq$ 2 des sommets du graphe de taille infinie associé à l'équation $\mathcal{E}_{\mathcal{C}}$.*

Proposition 8.19. *Sous l'hypothèse $\boldsymbol{\mathcal{H}}_1$ et l'hypothèse que ℓ est suffisamment grand pour que le graphe $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ contienne un sommet dont la taille du voisinage à distance $\leq$ 2 est égale à d_2, les graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ sont équivalents.*

Démonstration. De la même façon que pour la démonstration de la proposition 8.15, on montre que $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}})$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$ sont isomorphes. En reprenant les mêmes notations, l'isomorphisme $\psi : \tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}_{\mathcal{C}}}) \to \tilde{\mathcal{G}}(\mathcal{L}^1)$, $i \mapsto \pi(i) - sn$ permet de montrer l'équivalence. □

Ces graphes possèdent plus de sommets que les graphes $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ et $\mathcal{G}^1(\mathcal{L}^1)$ mais restent de petite taille comme le montre la proposition ci-dessous :

Proposition 8.20. *Soient $\mathcal{E}$ une équation de parité d'un code convolutif $(n, n-1)$ et s la longueur de $\mathcal{E}$. Le sous-graphe $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ contient au maximum $4\lceil \frac{s}{n} \rceil - 3$ sommets.*

Remarque. *Autrement dit, une équation de parité $\mathcal{E}$ de longueur s a, au maximum, $4\lceil \frac{s}{n} \rceil - 3$ équations dans son voisinage à distance 2.*

Démonstration. La démonstration de cette proposition est similaire à la preuve de la proposition 8.16. □

Remarque. *Avec l'utilisation de sous-graphes, l'hypothèse $\boldsymbol{\mathcal{H}}_1$ est allégée. En effet, pour retrouver l'équation de parité du code $\mathcal{D}$, il n'est pas nécessaire que $\mathcal{L}^1$ contienne ℓ équations qui, une fois désentrelacées, sont consécutives. Il suffit qu'il contienne une équation dont le voisinage à distance $\leq$ 2 dans $\mathcal{L}^1$ soit de taille d_2. Soit s la longueur de l'équation $\mathcal{E}_\mathcal{C}$, en utilisant la proposition 8.20, il s'en déduit que dans le pire des cas $\mathcal{L}^1$ doit contenir $4\lceil\frac{s}{n}\rceil - 3$ équations qui, une fois désentrelacées, sont consécutives.*

Avec ces différentes propositions, l'algorithme de recherche de l'équation de parité $\mathcal{E}_\mathcal{C}$ du code $\mathcal{D}$ devient l'algorithme 10.

Algorithme 10 : IDENTIFICATION_DU_CODE_CONVOLUTIF_2

Entrée :
$\mathcal{L}^1$ la liste des équations de parité de poids t d'une même classe d'équivalence
Sortie :
A une liste d'équations compatibles (avec l'assurance que $\mathcal{E}_\mathcal{C} \in A$)

$A \leftarrow \emptyset$
$\mathcal{E}^0 \leftarrow$ une équation de $\mathcal{L}^1$ dont la taille du voisinage à distance 2 est maximale
$\tilde{\mathcal{G}}(\mathcal{L}^1) \leftarrow$ le graphe associé à $\mathcal{L}^1$
$\mathcal{G}^1(\mathcal{L}^1) \leftarrow$ le sous-graphe de $\tilde{\mathcal{G}}(\mathcal{L}^1)$ induit par le voisinage à distance 1 de $\mathcal{E}^0$
$\tilde{\mathcal{G}}^2(\mathcal{L}^1) \leftarrow$ le sous-graphe de $\tilde{\mathcal{G}}(\mathcal{L}^1)$ induit par le voisinage à distance 2 de $\mathcal{E}^0$
Pour toutes les équations $\mathcal{E}$ de poids t et de longueur inférieure à s_{max}
  $\tilde{\mathcal{G}}(\mathcal{L}^\mathcal{E}) \leftarrow$ le graphe associé à $\mathcal{E}$
  $\mathcal{G}^1(\mathcal{L}^\mathcal{E}) \leftarrow$ le sous-graphe de $\tilde{\mathcal{G}}(\mathcal{L}^\mathcal{E})$ induit par le voisinage à distance 1 d'une équation de degré maximal
  Si $\mathcal{G}^1(\mathcal{L}^1)$ et $\mathcal{G}^1(\mathcal{L}^\mathcal{E})$ sont isomorphes
    $\tilde{\mathcal{G}}^2(\mathcal{L}^\mathcal{E}) \leftarrow$ le sous-graphe de $\tilde{\mathcal{G}}(\mathcal{L}^\mathcal{E})$ induit par le voisinage à distance 2 d'une équation dont la taille du voisinage à distance 2 est maximale
    Si $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^\mathcal{E})$ sont équivalents
      $A \leftarrow A \cup \{\mathcal{E}\}$
Retourner A

8.4.3 Réduction du nombre de tests

Les graphes et sous-graphes utilisés sont très fortement structurés et, comme expliqué précédemment, ils sont presque sommets-transitifs. Il n'est donc pas nécessaire de tester réellement toutes les équations de parité de poids t et de longueur inférieure à s_{max}. En effet, en utilisant la proposition suivante, du test d'une équation donnée il se déduit le résultat d'autres équations.

Proposition 8.21. *Soit l'équation de parité $\mathcal{E} = \{e_1, \ldots, e_t\}$ (avec $e_1 < e_2 < \cdots < e_t$) et s sa longueur. Sans perte de généralité, nous supposons que $e_1 = 1$. Cette équation peut également être représentée sous forme d'un vecteur binaire $b_1 \ldots b_s$ où $b_i = 1$ si $i \in \{e_1, \ldots, e_t\}$ et $b_i = 0$ sinon.*

— *Si $\mathcal{E}' = b_s b_{s-1} \ldots b_1$ (autrement dit, $\mathcal{E}'$ est le miroir de $\mathcal{E}$ par rapport à son support) alors les sous-graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^\mathcal{E})$ sont équivalents.*

— *Pour toute permutation $p = [p_1, \ldots, p_n]$ de taille n, on définit une permutation P par : $P(i) = \lfloor \frac{i-1}{n} \rfloor n + p_{[i]_n}$ où $[i]_n = i - \lfloor \frac{i-1}{n} \rfloor n$. Si l'équation $\mathcal{E}'$ est telle que $\mathcal{E}' = P(\mathcal{E})$ alors les sous-graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ sont équivalents.*

Démonstration. — Si $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ est formé des sommets représentant les équations $\mathcal{E}_{(i_1)}, \ldots, \mathcal{E}_{(i_m)}$ alors il existe un entier k tel que $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$ contient m sommets représentants : $\mathcal{E}'_{(i_1+k)}, \ldots, \mathcal{E}'_{(i_m+k)}$. L'isomorphisme $\phi : \tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}}) \to \tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$, $\mathcal{E}_{(i_j)} \mapsto \mathcal{E}'_{(i_{m-j}+k)}$ pour tout $j \in \{1, \ldots, m\}$ implique que les graphes sont isomorphes. Quant-à l'isomorphisme ψ, il implique l'équivalence de ces graphes : $\psi : \tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}}) \to \tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'}), i \mapsto x - i + 1$ où x est la plus grande étiquette du graphe $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$.

— Si $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ est formé des sommets représentant les équations $\mathcal{E}_{(i_1)}, \ldots, \mathcal{E}_{(i_m)}$ alors il existe un entier k tel que $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}'})$ contient m sommets représentants : $\mathcal{E}'_{(i_1+k)}, \ldots, \mathcal{E}'_{(i_m+k)}$. De plus $\#\{\mathcal{E}_{(i_a)} \cap \mathcal{E}_{(i_b)}\} = j$ si et seulement si $\#\{\mathcal{E}'_{(i_a+k)} \cap \mathcal{E}'_{(i_b+k)}\} = j$, les graphes sont donc isomorphes. La permutation P donne l'isomorphisme sur les étiquettes, ils sont donc équivalents.

□

Avec cette proposition, le nombre d'équations de parité à tester est considérablement réduit. Par exemple, si $n = 2$, $s_{max} = 20$ et $t = 10$, il suffit de tester 15 328 équations de parité alors qu'il en existe 184 756 de poids 10 et de longueur inférieure à 20. Ces exemples de paramètres n, s_{max} et t correspondent à ceux considérés lorsque le code $\mathcal{C}$ est défini par $(1 + D + D^2 + D^3 + D^6, 1 + D^2 + D^3 + D^5 + D^6)$. Dans ce cas, seules deux équations parmi les 15 328 testées passent le premier test sur l'isomorphisme des graphes $\mathcal{G}^1(\mathcal{L}^1)$ et $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$. Finalement, il n'en reste qu'une à la sortie de l'algorithme. De par la taille réduite des graphes $\mathcal{G}^1(\mathcal{L}^1)$ et $\mathcal{G}^1(\mathcal{L}^{\mathcal{E}})$ et de leur fort caractère discriminant, l'algorithme de recherche d'une équation de parité de $\mathcal{C}$ prend moins de 1 seconde.
Pour ce même exemple, si la valeur de s_{max} est fixée à 30, il est nécessaire de tester 1 238 380 équations au lieu de 30 045 015. Quatre graphes passent la première étape et deux équations sont retournées par l'algorithme, l'une d'elle est éliminée par la suite. Cette reconnaissance prend moins de 3 minutes.

Remarque. *Dans la proposition 8.21, deux équations conduisant à des graphes équivalents définissent des codes équivalents. Il est impossible de distinguer ces codes quelle que soit la taille du graphe considéré, mais surtout quelle que soit la méthode utilisée. Dans ces classes d'équivalence, une équation est choisie et un entrelaceur valide est reconstruit. De cet entrelaceur il se déduit les entrelaceurs employés avec les codes équivalents. Cette déduction se fait en appliquant les transformations nécessaires pour passer d'une équation à une autre sur l'entrelaceur.*

8.5 Ordonnancement des équations de parité

L'objectif de cette étape est de trouver un ordonnancement $\mathcal{A} = \mathcal{E}_{a_1}, \mathcal{E}_{a_2}, \ldots, \mathcal{E}_{a_\ell}$ des équations de $\mathcal{L}^1$ tel que $\pi^{-1}(\mathcal{E}_{a_i})$ soit égale à $\pi^{-1}(\mathcal{E}_{a_{i-1}})$ décalée de n positions vers la droite. Comme $\mathcal{L}^1$ contient uniquement des équations du même type, il faut que $\mathcal{A}$ contienne une et une seule fois chaque équation de $\mathcal{L}^1$.

L'isomorphisme entre les sommets des graphes $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$ donne exactement l'ordonnancement recherché.

Exemple. *En reprenant l'exemple des figures 8.7 et 8.8, l'isomorphisme ϕ entre les sommets de ces deux graphes est défini par : $\phi(\mathcal{E}^1) = \mathcal{E}'^3$, $\phi(\mathcal{E}^2) = \mathcal{E}'^1$, $\phi(\mathcal{E}^3) = \mathcal{E}'^4$ et $\phi(\mathcal{E}^4) = \mathcal{E}'^2$. Or, si la permutation $[\phi(1), \phi(2), \phi(3), \phi(4)] = [3, 1, 4, 2]$ est appliquée sur la numérotation des équations de la figure 8.8, autrement dit, si l'équation $\mathcal{E}'^3$ est placée en premier, $\mathcal{E}'^1$ en second,..., alors les équations sont "correctement" ordonnées.*

Ces graphes pouvant être de taille importante, l'isomorphisme est reconstruit pas à pas. Une partie de l'isomorphisme est déjà connue. En effet, pour déterminer si les sous-graphes $\tilde{\mathcal{G}}^2(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}^2(\mathcal{L}^{\mathcal{E}})$ sont équivalents il est nécessaire de retrouver l'isomorphisme entre les sommets de ces sous-graphes. C'est cette partie d'isomorphisme qui va être étendue pas à pas à l'ensemble des sommets de $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$.

Quelques notations sont nécessaires pour expliciter une étape de l'extension de ϕ :

Notation. *Le graphe contenant les sommets représentant les équations $\mathcal{E}_{(i)}$ pour tout entier $i \in [a, b]$ est noté $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^{\mathcal{E}})$. De façon similaire, l'isomorphisme ϕ défini sur toutes les équations $\mathcal{E}^i$ avec $i \in [a, b]$ est noté $\phi_{a..b}$. De plus, le graphe $\phi_{a..b}(\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^{\mathcal{E}}))$ est noté $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^1)$.*

Une étape d'extension se déroule comme suit :
Connaissant $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^{\mathcal{E}})$, $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^1)$ et $\phi_{a..b}$ il est recherché la valeur de $\phi(\mathcal{E}_{(b+1)})$ telle que $\tilde{\mathcal{G}}_{a..b+1}(\mathcal{L}^{\mathcal{E}})$ et $\tilde{\mathcal{G}}_{a..b+1}(\mathcal{L}^1)$ soient isomorphes.

- $\tilde{\mathcal{G}}_{a..b+1}(\mathcal{L}^{\mathcal{E}})$ est obtenu à partir de $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^{\mathcal{E}})$ en ajoutant le sommet représentant $\mathcal{E}_{(b+1)}$ ainsi que les arêtes correspondantes.
- Il est recherché l'équation $\mathcal{E}^i$ parmi l'ensemble des équations de $\mathcal{L}^1$ qui ne sont pas représentées dans $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^1)$ telle que si le sommet la représentant et les arêtes correspondantes sont ajoutées à $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^1)$ alors le graphe obtenu est isomorphe à $\tilde{\mathcal{G}}_{a..b+1}(\mathcal{L}^{\mathcal{E}})$. Une fois cette équation $\mathcal{E}^i$ trouvée, l'isomorphisme est étendu : $\phi_{a..b+1}(\mathcal{E}_{(j)}) = \phi_{a..b}(\mathcal{E}_{(j)})$ pour tout j compris entre a et b et $\phi_{a..b+1}(\mathcal{E}_{(b+1)}) = \mathcal{E}^i$.

Une fois qu'il n'est plus possible d'étendre l'isomorphisme du côté de b, le même raisonnement est effectué en recherchant la valeur de $\phi(\mathcal{E}_{(a-1)})$ à partir de $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^{\mathcal{E}})$, $\tilde{\mathcal{G}}_{a..b}(\mathcal{L}^1)$ et $\phi_{a..b}$.

Remarque. *En fonction de l'équation $\mathcal{E}$, il peut arriver que plusieurs équations de $\mathcal{L}^1$ soient compatibles pour étendre l'isomorphisme. Dans ce cas, une recherche en arbre est appliquée afin de retrouver l'isomorphisme impliquant toutes les équations de $\mathcal{L}^1$.*

Ce principe d'extension pas à pas de l'isomorphisme permet de surmonter le manque de certaines équations dans $\mathcal{L}^1$. En effet, il peut arriver que certaines équations soient manquantes dans $\mathcal{L}^1$, par exemple si l'une des équations n'a pas été classée dans $\mathcal{L}^1$ mais dans l'ensemble des équations non-classées, ou si l'algorithme de Dumer modifié n'a pas retrouvé toutes les équations de parité de poids t. Ces cas sont gérés en autorisant l'ajout dans $\tilde{\mathcal{G}}(\mathcal{L}^1)$ de sommets représentant des "équations manquantes" ainsi que les arêtes incidentes à ces sommets afin de conserver l'isomorphisme entre les graphes.

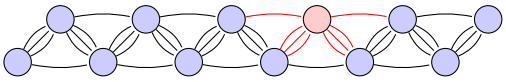


FIGURE 8.11 – Ajout d'une équation manquante

Remarque. *Le nombre maximal de sommets consécutifs représentant des "équations manquantes" est borné par* $\lfloor \frac{s}{n} \rfloor - 1$ *où* s *est la longueur de l'équation* $\mathcal{E}$*. En effet, si un nombre plus important de sommets consécutifs représentant des "équations manquantes" est ajouté au graphe, il n'y a aucun indice en commun entre la dernière équation non-manquante placée et l'équation recherchée.*

Exemple. *Sur la figure 8.11 est représenté un graphe étendu dans lequel un sommet représentant une équation manquante est ajouté (le sommet en rouge). Il y a tout de même une arête reliant le sommet précédant au sommet suivant de celui représentant l'équation manquante. Si un autre sommet manquant était ajouté consécutivement à celui-ci la connexion entre les sommets précédents et les sommets suivants serait perdue.*

Finalement, une fois l'isomorphisme entièrement reconstruit, il se déduit l'arrangement $\mathcal{A}$ souhaité.

8.6 Reconstruction de l'entrelaceur

Il faut désormais reconstruire l'entrelaceur π, cet entrelaceur ce déduit de l'isomorphisme entre les étiquettes de $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$.

Exemple. *En reprenant l'exemple illustré par les figures 8.8 et 8.9, un isomorphisme entre les étiquettes de ces deux graphes est défini par :* $\psi(4) = 1'$, $\psi(5) = 6'$, $\psi(6) = 9'$, $\psi(7) = 4'$, $\psi(8) = 12'$ *et* $\psi(10) = 5'$*. Or, si la permutation* $[\psi(1), \psi(2), \ldots, \psi(12)] = [?, ?, ?, 1, 6, 9, 4, 12, ?, 5, ?, ?]$ *est appliquée sur les colonnes du schéma représentant les équations de parité du code entrelacé, une partie du schéma des équations du code non-entrelacé est retrouvée. Les valeurs manquantes dans la définition de cet isomorphisme se déduisent des indices intervenant dans les équations de parité mais non représentés sur les graphes. La permutation définie par l'isomorphisme est égale à* π^{-1}*.*

Comme pour la recherche de l'isomorphisme ϕ entre les sommets des graphes $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$, l'isomorphisme ψ sur les étiquettes est reconstruit pas à pas. Pour cette reconstruction l'isomorphisme ϕ est utilisé, d'où l'intérêt de l'avoir recherché.

Pour retrouver ψ, il est recherché des sous-graphes $\tilde{\mathcal{G}}$ de $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$ où l'une des étiquettes, notée i, apparaît un nombre de fois différent des autres étiquettes. Il est alors recherché l'étiquette j apparaissant ce même nombre de fois dans le sous-graphe $\phi(\tilde{\mathcal{G}})$. Il s'en déduit $\psi(i) = j$.

Une fois que toutes les étiquettes de $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$ sont associées à une étiquette de $\tilde{\mathcal{G}}(\mathcal{L}^1)$, les derniers indices de ψ manquants se déduisent des indices intervenant dans les équations de parité mais qui ne sont pas représentés dans les graphes (de tels indices existent s'ils interviennent dans une unique équation de parité).

Finalement, il se déduit l'entrelaceur $\pi : \pi^{-1} = [\psi(min^{\mathcal{E}}), \psi(min^{\mathcal{E}} + 1), \ldots, \psi(max^{\mathcal{E}})]$ où $min^{\mathcal{E}}$ et $max^{\mathcal{E}}$ sont respectivement les indices minimaux et maximaux intervenant dans au moins une des équations de $\mathcal{L}^{\mathcal{E}}$.

Remarque. *Selon l'équation $\mathcal{E}$, il peut arriver qu'il y ait plusieurs isomorphismes entre les étiquettes des graphes $\tilde{\mathcal{G}}(\mathcal{L}^1)$ et $\tilde{\mathcal{G}}(\mathcal{L}^{\mathcal{E}})$. Dans ce cas, plusieurs entrelaceurs possibles sont déduits, les différences entre ces entrelaceurs portent uniquement sur les premières et les dernières positions. De plus, le nombre d'entrelaceurs reconstruits dépend uniquement de $\mathcal{E}$ et non de la longueur N de l'entrelaceur.*

8.7 Cas particuliers

Il peut arriver que certains cas particuliers se produisent, il est expliqué dans cette section comment les surmonter.

Présence d'indices indéterminés

Il peut arriver que certaines positions de l'entrelaceur ne soient pas retrouvées. Par exemple, lorsque plusieurs équations consécutives sont manquantes, certains indices n'interviendront alors dans aucune des équations retrouvées. Les équations n'apportent donc aucune information, il faut utiliser les mots de code bruités pour retrouver les valeurs manquantes.

Ces indices indéterminés sont, généralement, peu nombreux, il est donc envisageable de les tester de façon exhaustive. À l'aide de la permutation reconstruite jusque là, les indices manquants sont complétés afin de reconstituer les équations qui étaient manquantes. L'ensemble des mots de code bruités permettent de valider ou non les équations reconstituées.

Entrelaceur reconstruit de taille insuffisante

Si l'entrelaceur reconstruit est de taille inférieure à N, alors il faut étendre la partie reconstruite pour obtenir les positions manquantes. Par construction, ces positions sont nécessairement à ajouter au début ou à la fin. Cette situation se produit, par exemple, lorsque des équations du même type que celles contenues dans $\mathcal{L}^1$ sont placées dans $\mathcal{L}^{NC}$ (l'ensemble des équations "non-classées").

Pour retrouver ces positions, il est recherché, pas à pas, les équations de $\mathcal{L}^{NC}$ à ajouter en tête ou en fin de l'ordonnancement $\mathcal{A}$. À chaque prolongement, l'équation recherchée doit être compatible avec l'entrelaceur reconstruit jusque là, autant sur les tailles d'intersection avec les autres équations de l'ordonnancement, que sur les indices déjà positionnés. De telles équations sont très peu nombreuses, et généralement uniques.

8.8 Résultats pratiques

De nombreux tests ont été effectués afin d'évaluer les performances de cette méthode de reconstruction d'un code convolutif entrelacé. Quelques uns des résultats sont présentés et

N	M	temps équations	temps entrelaceur et code	temps total
1 000	400	2 sec	4 sec	6 sec
2 000	500	6 sec	4 sec	10 sec
5 000	1 400	40 sec	5 sec	45 sec
8 000	2 000	150 sec	7 sec	157 sec
10 000	2 600	300 sec	9 sec	309 sec

TABLE 8.1 – Temps nécessaires pour retrouver toutes les équations de parité de poids 8 puis pour la reconstruction de l'entrelaceur

commentés dans cette section.

Les temps présentés ci-dessous correspondent aux temps nécessaires pour reconnaître le code en utilisant la proposition 8.21, puis pour reconstruire tous les entrelaceurs pour chacun des codes passant l'étape de reconnaissance basée sur les équivalences de graphes.

8.8.1 Tests avec $\mathcal{C} = (1 + D + D^2 + D^5 + D^7, 1 + D + D^3 + D^4 + D^6)$

Le code convolutif $\mathcal{C}$ défini par $(1 + D + D^2 + D^5 + D^7, 1 + D + D^3 + D^4 + D^6)$ satisfait des équations de parité de poids 8. Toutes les équations de parité de ce poids sont du même type. Une fois les équations retrouvées par la méthode de Dumer modifiée, l'algorithme de reconstruction du code convolutif entrelacé est appliqué.

Pour ce code, une seule équation de parité est retournée par l'algorithme de reconnaissance du code convolutif (avec $s_{max} = 25$) et 16 entrelaceurs sont reconstruits pour cette équation. Ces entrelaceurs ne diffèrent, quelle que soit leur longueur, que sur les 4 premiers et les 4 derniers indices.

Les temps nécessaires pour les deux grandes étapes de la reconstruction sont indiqués pour différentes longueurs de permutation N dans la table 8.1. Il y est également inscrit le nombre M de mots entrelacés utilisés. Pour ces tests, les mots ne sont pas bruités, le bruit influençant uniquement la recherche des équations de petit poids et non la suite de la reconstruction du code entrelacé. En effet, une fois les équations de petit poids retrouvées, que les données soient bruitées ou non, le temps nécessaire pour finir la reconstruction est constant.

Avec cette table, il est mis en avant l'efficacité de la reconstruction de l'entrelaceur et de la reconnaissance du code, même lorsque l'entrelaceur est de taille importante. L'étape prenant le plus de temps est la recherche des équations de parité de petit poids par la méthode de Dumer modifiée.

Cependant, comme expliqué précédemment, il est possible de reconstruire l'entrelaceur et de retrouver le code même si toutes les équations de parité ne sont pas connues. Or, les dernières équations de parité sont les plus coûteuses à retrouver. Il est donc intéressant de pouvoir reconstruire l'entrelaceur sans avoir à retrouver toutes les équations. Dans la table 8.2 sont indiqués les temps nécessaires pour reconstruire l'entrelaceur lorsque qu'une partie des équations est manquante. Si le nombre d'équations manquantes est important alors la reconstruction de l'entrelaceur prend légèrement plus de temps, mais ce temps supplémentaire est très largement compensé par le temps gagné en ne recherchant pas les toutes dernières

N	M	nombre équations	temps équations	temps entrelaceur et code	temps total
2 000	500	990 (100%)	6 sec	4 sec	10 sec
		980 (99%)	4 sec	4 sec	8 sec
5 000	1 400	2490 (100%)	40 sec	5 sec	45 sec
		2480 (99.6%)	28 sec	5 sec	33 sec
		2470 (99.2%)	22 sec	6 sec	28 sec
8 000	2 000	3990 (100%)	150 sec	7 sec	157 sec
		3985 (99.9%)	130 sec	7 sec	137 sec
		3980 (99.7%)	120 sec	7 sec	127 sec
		3970 (99.5%)	100 sec	9 sec	109 sec
10 000	2 600	4990 (100%)	300 sec	9 sec	309 sec
		4980 (99.8%)	240 sec	9 sec	249 sec
		4970 (99.6%)	210 sec	9 sec	219 sec
		4960 (99.4%)	190 sec	10 sec	200 sec

TABLE 8.2 – Temps nécessaires pour reconstruire le code entrelacé, avec une partie des équations de parité, avec $\mathcal{C} = (1 + D + D^2 + D^5 + D^7, 1 + D + D^3 + D^4 + D^6)$

équations de parité. Par exemple, pour une permutation de longueur 10 000, le temps total passe alors de 309 secondes à 200 secondes. Plus l'entrelaceur est long, plus le gain sera significatif.

8.8.2 Tests avec $\mathcal{C} = (1 + D + D^2, 1 + D^2 + D^3)$

D'autres tests ont été effectués, en utilisant le code défini par $(1 + D + D^2, 1 + D^2 + D^3)$. Ce code satisfait des équations de poids 6 de 5 types différents. L'étape de classement des équations de parité est donc indispensable pour ce code. Les temps nécessaires pour retrouver les équations de parité puis reconstruire l'entrelaceur sont reportés dans la table 8.3, et pour ces tests $s_{max} = 10$.

Avec ce code, il est mis en avant le fait que le code peut être retrouvé sans pouvoir reconstruire entièrement la permutation. Par exemple, lorsque $N = 8\,000$, un ensemble de 19 240 équations de parité (soit 96.3% des équations) est suffisant pour retrouver le code, mais la permutation ne peut être reconstruite.

Le cas bruité a également été testé, par exemple pour un entrelaceur de longueur $N = 1\,000$ avec un ensemble de 200 mots bruités par un canal binaire symétrique de probabilité $\tau = 10^{-4}$, il faut environ 150 secondes pour retrouver toutes les équations de parité de poids 6. Cependant, les équations retrouvées en 30 secondes sont suffisantes pour reconstruire la permutation, ainsi le temps total est réduit à 31 secondes au lieu de 151 secondes. Comme pour cet exemple, pouvoir reconstruire l'entrelaceur sans posséder toutes les équations de parité permet de réduire de façon encore plus importante le temps total de la reconstruction lorsque les données sont bruitées.

N	M	nombre équations	temps équations	temps entrelaceur et code	temps total
1 000	200	2 475 (100%)	1 sec	0.1 sec	1.1 sec
2 000	400	4 475 (100%)	3 sec	0.4 sec	3.4 sec
5 000	900	12 475 (100%)	15 sec	3 sec	18 sec
		12 437 (99.7%)	10 sec	3 sec	13 sec
8 000	1 300	19 975 (100%)	60 sec	7 sec	67 sec
		19 940 (99.8%)	40 sec	7 sec	47 sec
10 000	1 700	24 975 (100%)	130 sec	11 sec	141 sec
		24 970 (99.9%)	100 sec	11 sec	111 sec

TABLE 8.3 – Temps nécessaires pour reconstruire le code entrelacé, avec une partie des équations de parité, avec $\mathcal{C} = (1 + D + D^2, 1 + D^2 + D^3)$

8.8.3 Tests avec $\mathcal{C} = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$

La reconnaissance du code $\mathcal{C} = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$ entrelacé a été testée. Ce code satisfait 11 types d'équations de parité de poids 10. Afin de retrouver le code entrelacé, il est donc nécessaire de retrouver de très nombreuses équations de parité. Sur cet exemple, pouvoir reconstruire le code et l'entrelaceur sans disposer de toutes les équations de parité permet de réduire significativement le temps total de la reconstruction, comme noté dans la table 8.4. Pour ces tests, la longueur maximale de l'équation du code $\mathcal{D}$ est fixée à $s_{max} = 20$.

Le nombre minimal d'équations à retrouver afin de pouvoir reconstruire l'entrelaceur n'est pas fixé, il dépend des équations retrouvées par l'algorithme de Dumer modifié. Par exemple, pour $N = 10\,000$, 53 240 équations sont retrouvées en 760 secondes (soit 97% des équations), il est arrivé que l'entrelaceur soit correctement reconstruit avec cet ensemble d'équations, alors que dans certains cas en posséder 53 900 n'est pas suffisant.

N	M	nombre équations	temps équations	temps entrelaceur et code	temps total
1 000	400	5 390 (100%)	7 sec	2 sec	9 sec
		5 375 (99.7%)	4 sec	2 sec	6 sec
2 000	600	10 890 (100%)	35 sec	5 sec	40 sec
		10 880 (99.9%)	25 sec	5 sec	30 sec
		10 845 (99.6%)	18 sec	6 sec	24 sec
		10 745 (98.7%)	14 sec	6 sec	20 sec
		10 610 (97.4%)	12 sec	6 sec	18 sec
5 000	1 600	27 390 (100%)	250 sec	29 sec	279 sec
		27 380 (99.9%)	190 sec	29 sec	219 sec
		27 360 (99.9%)	155 sec	29 sec	184 sec
		27 320 (99.7%)	135 sec	29 sec	164 sec
8 000	2 400	43 890 (100%)	1 100 sec	74 sec	1 174 sec
		43 850 (99.9%)	640 sec	74 sec	714 sec
		43 700 (99.6%)	500 sec	74 sec	574 sec
		43 220 (98.5%)	420 sec	74 sec	494 sec
10 000	2 800	54 890 (100%)	2 200 sec	115 sec	2 315 sec
		54 850 (98.9%)	1 630 sec	115 sec	1 745 sec
		54 000 (98.4%)	900 sec	115 sec	1 015 sec

TABLE 8.4 – Temps nécessaires pour reconstruire le code entrelacé, avec une partie des équations de parité, avec $\mathcal{C} = (1 + D^2 + D^3 + D^5 + D^6, 1 + D + D^2 + D^3 + D^6)$

Conclusion et perspectives

Trois nouvelles méthodes de reconnaissance de codes correcteurs ont été présentées. Elles ont été testées à partir de données bruitées afin de mettre en évidence leurs capacités et leurs limites.

Reconnaissance des turbo-codes

La première méthode proposée s'intéresse à la reconstruction de la permutation interne des turbo-codes. Cette méthode est applicable aux différents types de turbo-codes : les turbo-codes parallèles, qu'ils soient ou non poinçonnés, ainsi qu'aux turbo-codes série. Dans tous les cas, il est nécessaire de connaître le code convolutif utilisé, l'initialisation des registres au début du codage de chacun des mots ainsi que le motif de poinçonnage. Même si l'hypothèse sur la connaissance du code convolutif peut être levée, les deux autres sont indispensables. Cependant, elles sont généralement satisfaites.

Il a été mis en évidence les capacités de cette méthode de reconstruction, qui retrouve rapidement la permutation même pour des longueurs importantes (par exemple 10 000) ou lorsque le niveau de bruit dépasse largement la capacité de correction du code.

La limite de cette méthode porte sur le motif de poinçonnage, en effet, la complexité est exponentielle en le nombre de valeurs poinçonnées consécutivement. Or, en pratique il arrive que ce nombre soit trop important. Par exemple, dans la norme ETSI GMR-1 [47] afin d'obtenir un turbo-code de rendement 5/6, 9/10-ième des bits du vecteur de redondance $\boldsymbol{w}$ sont poinçonnés, il est alors inenvisageable d'utiliser la méthode proposée (même si les données ne sont pas bruitées).

Reconnaissance des codes LDPC

Nous nous sommes ensuite intéressé à la reconnaissance des codes LDPC. Pour cela, la méthode de Dumer de recherche de mots de petits poids a été adaptée afin d'utiliser les informations connues sur la répartition de poids des mots recherchés. Avec cette nouvelle méthode, les méthodes existantes sont généralisées. Les tests effectués ont montré leur efficacité et la faible quantité de données nécessaire, en particulier lorsque les données sont bruitées. Même si la méthode est efficace, elle atteint ses limites lorsque le poids des équations de parité à retrouver est élevé et que le code est de longueur importante. Par exemple, la reconnaissance du code LDPC de la norme DVB-S2 [42] de longueur 64 200 satisfaisant des équations de parité de poids 30 serait extrêmement coûteux voire, en fonction du niveau de bruit, infaisable.

L'étude des codes LDPC normés (dans le chapitre 4 a permis de mettre en évidence les structures quasi-cycliques (ou presque quasi-cycliques) des matrices de parité utilisées en pratique. Il serait intéressant d'utiliser ces structures afin d'accélérer leurs reconstructions et ainsi pouvoir repousser les limites de la reconnaissance. Lorsque la taille des sous-matrices cycliques n'est pas un nombre premier, une des idées envisageables est d'utiliser la méthode proposée dans [90]. Avec cette méthode, les équations de parité sont recherchées dans un code plus petit, en contrepartie, le niveau de bruit est augmenté. Remarquons, que dans les codes LDPC normés, la taille des sous-matrices cycliques est très rarement un nombre premier.

Reconnaissance des codes convolutifs entrelacés

La méthode de recherche d'équations de parité de petit poids est particulièrement bien adaptée à la reconnaissance des codes LDPC mais pas uniquement. En effet, celle-ci peut être utilisée pour reconstruire d'autres familles de code. C'est ce qui est effectué, dans le dernier chapitre, pour la reconnaissance des codes convolutifs entrelacés.

Une nouvelle méthode de reconstruction des codes convolutifs entrelacés a donc été présentée. L'intérêt de cette méthode est qu'elle reconstruit l'entrelaceur sans aucune hypothèse sur sa structure contrairement aux méthodes existantes. De plus, un sous-code du code convolutif est reconnu dans le même temps, si le code utilisé n'est pas de type $(n, n-1)$ alors, une fois l'entrelaceur reconstruit, les données peuvent être désentrelacées, et une des méthodes classiques de reconnaissance des codes convolutifs est appliquée afin de le retrouver. Ce raisonnement peut être appliqué à tous les codes convolutifs et même s'ils sont poinçonnés à condition que le motif de poinçonnage soit périodique. Notons, que la connaissance de ce motif n'est pas nécessaire lors de la reconstruction. Les tests effectués ont montré l'efficacité de cette méthode même sur des entrelaceurs de grande longueur. Les limites de cette méthode sont les mêmes que pour la reconnaissance des codes LDPC, en effet, l'étape la plus coûteuse est la recherche des équations de parité de petit poids.

Une fois les équations retrouvées, la suite de la reconnaissance fait appel à la théorie des graphes. La simplicité de la méthode se reflète dans les temps nécessaires à la reconstruction de l'entrelaceur une fois la liste des équations de parité retrouvée. En effet, dans tous les tests effectués, il faut moins de 2 minutes pour retrouver un sous-code du code convolutif et reconstruire l'entrelaceur.

Un autre intérêt de cette méthode est qu'elle est applicable à la reconnaissance des turbo-codes. En effet, la reconstruction de la permutation interne d'un turbo-code est le même problème que celui de la reconnaissance des codes convolutifs entrelacés. Plus précisément, pour les turbo-codes série, la problématique est exactement la même, la seule différence porte sur les hypothèses ; pour la méthode du chapitre 5 spécifique aux turbo-codes, le codeur convolutif est supposé connu, tout comme l'initialisation de sa mémoire. Ces hypothèses ne sont plus nécessaires pour la méthode du chapitre 8. La permutation interne des turbo-codes peut donc être reconstruite dans des cas non-accessibles à l'algorithme présenté dans le chapitre 5.
C'est également le cas pour les turbo-codes parallèles. Dans ce cas, les hypothèses faites dans le chapitre 5 portent, en plus, sur la séparation des sorties du turbo-code et utilise très largement le fait que la sortie de redondance est ordonnée. Rapporté au problème de la reconnaissance

d'un code convolutif entrelacé, ces hypothèses correspondent à la connaissance d'une partie de l'entrelaceur recherché. La méthode de reconnaissance des codes convolutifs entrelacés peut intégrer ces informations supplémentaires afin de reconstruire plus efficacement l'entrelaceur et permet de ne plus supposer connu le codeur convolutif, ni l'initialisation de sa mémoire, ni le motif de poinçonnage. Toutes ces informations étant indispensable à la méthode du chapitre 5. De plus, le motif de poinçonnage est une des limites de la méthode initiale de par le nombre de bits poinçonnés consécutivement, ce n'est plus le cas avec la dernière méthode présentée. Cependant le poids des équations de parité à rechercher sera tout de même plus important.

Annexe A

Les codes LDPC binaires normés

Pour chaque structure de code LDPC, les paramètres utilisés dans les différentes normes sont regroupés ci-dessous. La longueur N, la dimension K, le rendement R ainsi que, le cas échéant, la taille des sous-matrices cycliques Z sont donnés. Le polynôme $\rho(x)$ déterminant la répartition de poids des lignes de la matrice de parité du code considéré est également explicité. Ce polynôme est défini de la façon suivante :
$\rho(x) = \sum_i \rho_i x^i$ où ρ_i est la proportion de lignes de poids i dans la matrice de parité $\boldsymbol{\mathcal{H}}$.

A.1 Codes LDPC avec une structure convolutive

A.1.1 Codes LDPC quasi-cycliques avec structure convolutive

a. Normes IEEE 802.11*, 802.16*, 802.22 et WiMedia

Pour rappel, la notation 802.11* regroupe les normes IEEE 802.11, 802.11n, 802.11ac et 802.11af. La notation 802.16* représente les normes 802.16 et 802.16e.

N	K	R	Z	Normes
2 304	1 152	1/2	96	802.16*, 802.22
2 304	1 536	2/3	96	802.16*, 802.22
2 304	1 728	3/4	96	802.16*, 802.22
2 304	1 920	5/6	96	802.16*, 802.22
2 208	1 104	1/2	92	802.16*, 802.22
2 208	1 472	2/3	92	802.16*, 802.22
2 208	1 656	3/4	92	802.16*, 802.22
2 208	1 840	5/6	92	802.16*, 802.22
2 112	1 056	1/2	88	802.16*, 802.22
2 112	1 408	2/3	88	802.16*, 802.22
2 112	1 584	3/4	88	802.16*, 802.22
2 112	1 760	5/6	88	802.16*, 802.22

N	K	R	Z	Normes
2 016	1 008	1/2	84	802.16*, 802.22
2 016	1 344	2/3	84	802.16*, 802.22
2 016	1 512	3/4	84	802.16*, 802.22
2 016	1 680	5/6	84	802.16*, 802.22
1 944	972	1/2	81	802.11*
1 944	1 296	2/3	81	802.11*
1 944	1 458	3/4	81	802.11*
1 944	1 620	5/6	81	802.11*
1 920	960	1/2	80	802.16*, 802.22
1 920	1 280	2/3	80	802.16*, 802.22
1 920	1 440	3/4	80	802.16*, 802.22
1 920	1 600	5/6	80	802.16*, 802.22

N	K	R	Z	Normes
1 824	912	1/2	76	802.16*, 802.22
1 824	1 216	2/3	76	802.16*, 802.22
1 824	1 368	3/4	76	802.16*, 802.22
1 824	1 520	5/6	76	802.16*, 802.22
1 728	864	1/2	72	802.16*, 802.22
1 728	1 152	2/3	72	802.16*, 802.22
1 728	1 296	3/4	72	802.16*, 802.22
1 728	1 440	5/6	72	802.16*, 802.22
1 632	816	1/2	68	802.16*, 802.22
1 632	1 088	2/3	68	802.16*, 802.22
1 632	1 224	3/4	64	802.16*, 802.22
1 632	1 360	5/6	64	802.16*, 802.22
1 536	768	1/2	64	802.16*, 802.22
1 536	1 024	2/3	64	802.16*, 802.22
1 536	1 152	3/4	64	802.16*, 802.22
1 536	1 280	5/6	64	802.16*, 802.22
1 440	720	1/2	60	802.16*, 802.22
1 440	960	2/3	60	802.16*, 802.22
1 440	1 080	3/4	60	802.16*, 802.22
1 440	1 200	5/6	60	802.16*, 802.22
1 344	672	1/2	56	802.16*, 802.22
1 344	896	2/3	56	802.16*, 802.22
1 344	1 008	3/4	56	802.16*, 802.22
1 344	1 120	5/6	56	802.16*, 802.22
1 296	648	1/2	54	802.11*
1 296	864	2/3	54	802.11*
1 296	972	3/4	54	802.11*
1 296	1 080	5/6	54	802.11*
1 248	642	1/2	52	802.16*, 802.22
1 248	832	2/3	52	802.16*, 802.22
1 248	936	3/4	52	802.16*, 802.22
1 248	1 040	5/6	52	802.16*, 802.22
1 200	600	1/2	30	WiMedia
1 200	750	5/8	30	WiMedia
1 200	900	3/4	30	WiMedia
1 200	960	4/5	30	WiMedia
1 152	576	1/2	48	802.16*, 802.22
1 152	768	2/3	48	802.16*, 802.22
1 152	864	3/4	48	802.16*, 802.22
1 152	960	5/6	48	802.16*, 802.22
1 056	528	1/2	44	802.16*, 802.22
1 056	704	2/3	44	802.16*, 802.22
1 056	792	3/4	44	802.16*, 802.22
1 056	880	5/6	44	802.16*, 802.22
960	480	1/2	40	802.16*, 802.22
960	640	2/3	40	802.16*, 802.22
960	720	3/4	40	802.16*, 802.22
960	800	5/6	40	802.16*, 802.22
864	432	1/2	36	802.16*, 802.22
864	576	2/3	36	802.16*, 802.22
864	648	3/4	36	802.16*, 802.22
864	720	5/6	36	802.16*, 802.22
768	384	1/2	32	802.16*, 802.22
768	512	2/3	32	802.16*, 802.22
768	576	3/4	32	802.16*, 802.22
768	640	5/6	32	802.16*, 802.22
672	336	1/2	28	802.16*, 802.22
672	448	2/3	28	802.16*, 802.22
672	504	3/4	28	802.16*, 802.22
672	560	5/6	28	802.16*, 802.22
648	324	1/2	27	802.11*
648	432	2/3	27	802.11*
648	486	3/4	27	802.11*
648	540	5/6	27	802.11*
576	288	1/2	24	802.16*, 802.22
576	384	2/3	24	802.16*, 802.22
576	432	3/4	24	802.16*, 802.22
576	480	5/6	24	802.16*, 802.22
480	240	1/2	20	802.22
480	320	2/3	20	802.22
480	360	3/4	20	802.22
480	400	5/6	20	802.22
384	192	1/2	16	802.22
384	256	2/3	16	802.22
384	288	3/4	16	802.22
384	320	5/6	16	802.22

- Norme 802.11*
  - Pour $R = 1/2$
    - avec $N = 648$ $\rho(x) = \frac{2}{3}x^7 + \frac{1}{3}x^8$
    - avec $N = 1\,296$ ou $1\,944$ $\rho(x) = \frac{5}{6}x^7 + \frac{1}{6}x^8$
  - Pour $R = 2/3$
    - avec $N = 648, 1\,296$ ou $1\,944$ $\rho(x) = x^{11}$

  - Pour $R = 3/4$
    - avec $N = 648$ ou $1\,296$ $\rho(x) = \frac{1}{3}x^{14} + \frac{2}{3}x^{15}$
    - avec $N = 1\,944$ $\rho(x) = \frac{5}{6}x^{14} + \frac{1}{6}x^{15}$
  - Pour $R = 5/6$
    - avec $N = 648$ $\rho(x) = x^{22}$
    - avec $N = 1\,296$ $\rho(x) = \frac{3}{4}x^{21} + \frac{1}{4}x^{22}$
    - avec $N = 1\,944$ $\rho(x) = \frac{1}{5}x^{19} + \frac{4}{5}x^{20}$

- Normes 802.16* et 802.22
  - Pour $R = 1/2$
    - (norme 802.16* et 802.22) $\rho(x) = \frac{2}{3}x^{6} + \frac{1}{3}x^{7}$
  - Pour $R = 2/3$
    - (normes 802.16* et 802.22) $\rho(x) = \frac{7}{8}x^{10} + \frac{1}{8}x^{11}$
    - (norme 802.16*) $\rho(x) = x^{10}$
  - Pour $R = 3/4$
    - (normes 802.16* et 802.22) $\rho(x) = \frac{5}{6}x^{14} + \frac{1}{6}x^{15}$
    - (norme 802.16*) $\rho(x) = \frac{1}{3}x^{14} + \frac{2}{3}x^{15}$
  - Pour $R = 5/6$
    - (norme 802.16* et 802.22) $\rho(x) = x^{20}$

- Norme WiMedia
  - Pour $R = 1/2$ $\rho(x) = \frac{3}{20}x^{5} + \frac{17}{20}x^{6}$
  - Pour $R = 5/8$ $\rho(x) = \frac{11}{15}x^{8} + \frac{4}{15}x^{9}$
  - Pour $R = 3/4$ $\rho(x) = x^{12}$
  - Pour $R = 4/5$ $\rho(x) = \frac{1}{8}x^{12} + \frac{7}{8}x^{15}$

b. Norme WiMedia (deuxième structure)

N	K	R	Z	Normes
1 320	600	5/11	30	WiMedia
1 320	750	25/44	30	WiMedia
1 320	900	15/22	30	WiMedia
1 320	960	8/11	30	WiMedia

  - Pour $R = 5/11$ $\rho(x) = \frac{3}{24}x^{5} + \frac{21}{24}x^{6}$
  - Pour $R = 25/44$ $\rho(x) = \frac{12}{19}x^{8} + \frac{7}{19}x^{9}$
  - Pour $R = 15/22$ $\rho(x) = \frac{1}{14}(x^{7} + x^{8} + x^{9} + x^{11}) + \frac{5}{7}x^{12}$
  - Pour $R = 8/11$ $\rho(x) = \frac{1}{12}(x^{7} + x^{9}) + \frac{1}{6}x^{11} + \frac{1}{12}x^{12} + \frac{7}{12}x^{15}$

c. Norme IUT-G.9960

N	K	R	Z	Normes
336	168	1/2	14	G.9960
1 152	960	5/6	48	G.9960
1 440	960	2/3	60	G.9960
1 920	960	1/2	80	G.9960
5 184	4 320	5/6	216	G.9960
6 480	4 320	2/3	270	G.9960
8 640	4 320	1/2	360	G.9960

— Pour $R = 1/2$ $\rho(x) = \frac{1}{12}x^5 + \frac{5}{12}x^6 + \frac{1}{2}x^7$
— Pour $R = 2/3$ $\rho(x) = \frac{1}{8}x^9 + \frac{1}{2}x^{10} + \frac{3}{8}x^{11}$
— Pour $R = 5/6$ $\rho(x) = \frac{1}{4}x^{19} + \frac{1}{2}x^{20} + \frac{1}{2}x^{21}$

d. Norme IEEE 802.20*

Par 802.20* sont représentées les normes IEEE 802.20 et 802.20.2. Dans ces normes, les tailles sont variables et dépendent d'un entier L.

N	K	R	Z	Normes
33L	6L	2/11	L	802.20*
38L	7L	7/38	L	802.20*
43L	8L	8/43	L	802.20*
48L	9L	9/48	L	802.20*
53L	10L	10/53	L	802.20*
58L	11L	11/58	L	802.20*

— Pour $R = 2/11$ $\rho(x) = \frac{1}{3}x^3 + \frac{5}{9}x^4 + \frac{1}{9}x^6$
— Pour $R = 7/38$ $\rho(x) = \frac{9}{31}x^3 + \frac{18}{31}x^4 + \frac{3}{31}x^5 + \frac{1}{31}x^6$
— Pour $R = 8/43$ $\rho(x) = \frac{9}{35}x^3 + \frac{4}{7}x^4 + \frac{4}{35}x^5 + \frac{2}{35}x^7$
— Pour $R = 3/16$ $\rho(x) = \frac{10}{39}x^3 + \frac{20}{39}x^4 + \frac{3}{13}x^5$
— Pour $R = 10/53$ $\rho(x) = \frac{9}{43}x^3 + \frac{25}{43}x^4 + \frac{6}{43}x^5 + \frac{3}{43}x^6$
— Pour $R = 11/58$ $\rho(x) = \frac{11}{47}x^3 + \frac{27}{47}x^4 + \frac{3}{47}x^5 + \frac{6}{47}x^6$

A.1.2 Codes LDPC avec structure convolutive presque quasi-cycliques après permutation

Afin de simplifier les coefficients des polynôme $\rho(x)$, deux notations sont utilisées :

$$a = \frac{1}{N-K} \quad \text{et} \quad b = \frac{N-K-1}{N-K} = 1 - a$$

N	K	R	Z	Normes	$\rho(x)$
64 800	14 400	2/9	360	DVB-S2X	$ax^3 + bx^4$
64 800	16 200	1/4	360	DVB-S2	$ax^3 + bx^4$
64 800	18 720	13/45	360	DVB-S2X	$0.09x^4 + 0.9x^5$
64 800	21 600	1/3	360	DVB-S2	$ax^4 + bx^5$
64 800	25 920	2/5	360	DVB-S2	$ax^5 + bx^6$
64 800	29 160	9/20	360	DVB-S2X	$ax^6 + bx^7$
64 800	32 400	1/2	360	DVB-S2, T2	$ax^6 + bx^7$
64 800	32 400	1/2	360	DVB-S2X	$ax^6 + 0.11x^7 + 0.89x^8$
64 800	34 560	8/15	360	DVB-S2X	$ax^7 + 0.18x^8 + 0.82x^9$
64 800	35 640	11/20	360	DVB-S2X	$ax^8 + bx^9$
64 800	36 000	5/9	360	DVB-S2X	$ax^7 + 0.12x^8 + 0.87x^9$
64 800	37 440	26/45	360	DVB-S2X	$ax^8 + 0.08x^9 + 0.92x^{10}$
64 800	37 440	26/45	360	DVB-S2X	$ax^9 + bx^{10}$
64 800	38 880	3/5	360	DVB-S2, T2	$ax^{10} + bx^{11}$
64 800	38 880	3/5	360	DVB-S2X	$0.01x^{10} + 0.99x^{11}$
64 800	40 320	28/45	360	DVB-S2X	$ax^9 + bx^{10}$
64 800	41 400	23/36	360	DVB-S2X	$ax^9 + bx^{10}$
64 800	41 760	29/45	360	DVB-S2X	$ax^{10} + 0.27x^{11} + 0.73x^{12}$
64 800	43 200	2/3	360	DVB-S2, T2, C2	$ax^9 + bx^{10}$
64 800	43 200	2/3	360	DVB-S2X	$ax^{12} + 0.98x^{13} + 0.02x^{14}$
64 800	44 640	31/45	360	DVB-S2X	$ax^{12} + 0.45x^{13} + 0.55x^{14}$
64 800	45 000	25/36	360	DVB-S2X	$ax^{12} + bx^{13}$
64 800	46 080	32/45	360	DVB-S2X	$ax^{13} + 0.52x^{14} + 0.48x^{12}$
64 800	46 800	13/18	360	DVB-S2X	$ax^{13} + bx^{14}$
64 800	47 520	11/15	360	DVB-S2X	$ax^{14} + 0.23x^{15} + 0.77x^{16}$
64 800	47 520	11/15	360	DVB-S2X	$ax^{15} + 0.85x^{16} + 0.15x^{17}$
64 800	48 600	3/4	360	DVB-S2, T2, C2	$ax^{13} + bx^{14}$
64 800	48 600	3/4	360	DVB-S2X	$ax^{15} + 0.67x^{16} + 0.33x^{17}$
64 800	50 400	7/9	360	DVB-S2X	$ax^{18} + 0.37x^{19} + 0.63x^{20}$
64 800	50 400	7/9	360	DVB-S2X	$ax^{17} + bx^{18}$
64 800	51 840	4/5	360	DVB-S2, T2, C2	$ax^{17} + bx^{18}$
64 800	54 000	5/6	360	DVB-S2, T2, C2	$ax^{21} + bx^{22}$
64 800	55 440	77/90	360	DVB-S2X	$ax^{28} + 0.81x^{29} + 0.19x^{30}$
64 800	57 600	8/9	360	DVB-S2	$ax^{26} + bx^{27}$
64 800	58 320	9/10	360	DVB-S2, C2	$ax^{29} + bx^{30}$
32 400	6 480	1/59	360	DVB-S2X	$ax^3 + bx^4$
32 400	7 920	11/45	360	DVB-S2X	$ax^3 + bx^4$
32 400	10 800	1/3	360	DVB-S2X	$ax^4 + bx^5$
16 200	3 240	1/5	360	DVB-S2, T2, NGH	$0.25x^3 + 0.75x^4$
16 200	3 960	11/45	360	DVB-S2X	$ax^3 + bx^4$
16 200	4 320	4/15	360	DVB-S2, NGH	$0.09x^4 + 0.91x^5$
16 200	5 040	14/45	360	DVB-S2X	$ax^4 + bx^5$
16 200	5 400	1/3	360	DVB-S2, T2, NGH	$ax^4 + bx^5$
16 200	6 480	2/5	360	DVB-S2, T2, NGH	$ax^5 + bx^6$
16 200	7 200	4/9	360	DVB-S2, T2, C2	$0.16x^4 + 0.36x^5 + 0.4x^6 + 0.08x^7$
16 200	7 560	7/15	360	DVB-S2X, NGH	$0.54x^8 + 0.46x^4$
16 200	8 640	8/15	360	DVB-S2X, NGH	$0.05x^9 + 0.95x^{10}$
16 200	9 360	26/45	360	DVB-S2X	$ax^9 + bx^{10}$
16 200	9 720	3/5	360	DVB-S2	$ax^{10} + bx^{11}$

N	K	R	Z	Normes	$\rho(x)$
16 200	9 720	3/5	360	DVB-T2, NGH	$ax^8 + bx^9$
16 200	10 800	2/3	360	DVB-S2, T2, C2, NGH	$ax^9 + bx^{10}$
16 200	11 520	32/45	360	DVB-S2X	$ax^{12} + bx^{13}$
16 200	11 880	11/15	360	DVB-S2, T2, C2, NGH	$0.08x^9 + 0.25x^{10} + 0.33x^{11} + 0.25x^{12} + 0.08x^{13}$
16 200	12 600	7/9	360	DVB-S2, T2, C2	$0.1x^{11} + 0.3x^{12} + 0.6x^{13}$
16 200	13 320	37/45	360	DVB-S2, T2, C2	$ax^{15} + 0.5x^{16} + 0.12x^{17} + 0.12x^{18} + 0.25x^{19}$
16 200	14 400	8/9	360	DVB-S2, C2	$ax^{26} + bx^{27}$
11 136	8 352	3/4	87	GMR-1, GMPRS-1	$ax^{14} + bx^{15}$
8 880	5 920	2/3	74	GMR-1, GMPRS-1	$ax^{10} + bx^{11}$
8 880	11 100	4/5	74	GMR-1, GMPRS-1	$ax^{19} + bx^{20}$
8 640	2 160	1/4	72	DVN-NGH	$0.01x^2 + 0.01x^3 + 0.14x^4 + \cdots + 0.07x^9 + 0.02x^{10}$
4 480	4 032	9/10	74	GMR-1, GMPRS-1	$ax^{36} + bx^{37}$
4 464	2 232	1/2	62	GMR-1, GMPRS-1	$ax^6 + bx^7$
4 440	2 960	2/3	74	GMR-1, GMPRS-1	$ax^{10} + bx^{11}$
4 440	3 552	4/5	74	GMR-1, GMPRS-1	$ax^{19} + bx^{20}$
4 320	864	1/5	72	DVB-NGH	$ax^2 + 0.44x^3 + 0.56x^4$
4 320	2 160	1/2	72	DVB-NGH	$0.5x^8 + 0.5x^9$
4 096	1 024	1/4	128	DVB-RCS+M	$ax^3 + bx^4$
4 096	2 048	1/2	128	DVB-RCS+M	$ax^6 + bx^7$
4 096	3 072	3/4	128	DVB-RCS+M	$ax^{16} + bx^{17}$
2 400	1 800	3/4	50	GMR-1, GMPRS-1	$ax^{14} + bx^{15}$
2 400	1 920	4/5	50	GMR-1, GMPRS-1	$ax^{19} + bx^{20}$
1 920	1 536	4/5	32	GMR-1, GMPRS-1	$ax^{19} + bx^{20}$
1 920	1 728	9/10	32	GMR-1, GMPRS-1	$ax^{36} + bx^{37}$
1 908	1 272	2/3	53	GMR-1, GMPRS-1	$ax^{11} + bx^{12}$
976	488	1/2	61	GMR-1, GMPRS-1	$ax^6 + bx^7$
960	640	2/3	32	GMR-1, GMPRS-1	$ax^{10} + bx^{11}$
960	864	9/10	16	GMR-1, GMPRS-1	$ax^{31} + bx^{32}$
950	760	4/5	19	GMR-1, GMPRS-1	$ax^{19} + bx^{20}$

N	K	R	Z	Normes	$\rho(x)$
24 576	16 384	2/3	32	CCSDS 131.5	$0.19x^{10} + 0.81x^{11}$
20 480	16 384	4/5	16	CCSDS 131.5	$ax^{20} + bx^{21}$
18 432	16 384	8/9	16	CCSDS 131.5	$ax^{33} + bx^{34}$
3 072	2 048	2/3	8	CCSDS 131.5	$0.5x^{10} + ax^{11} + 0.499x^{12}$
2 560	2 048	4/5	8	CCSDS 131.5	$ax^{19} + bx^{20}$
2 304	2 048	8/9	8	CCSDS 131.5	$0.5x^{50} + 0.5x^{51}$
768	512	2/3	8	CCSDS 131.5	$ax^{11} + bx^{12}$
640	512	4/5	8	CCSDS 131.5	$ax^{21} + bx^{22}$
576	512	8/9	8	CCSDS 131.5	$ax^{55} + bx^{51}$

A.2 Codes LDPC sans structure convolutive

A.2.1 Codes LDPC quasi-cycliques sans structure convolutive

a. Normes IEEE 802.11ad et 802.15.3c

N	K	R	Z	Normes
672	336	1/2	42	802.11ad
672	420	5/8	42	802.11ad
672	420	5/8	21	802.15.3c
672	504	3/4	42	802.11ad
672	504	3/4	21	802.15.3c
672	546	13/16	42	802.11ad
672	588	7/8	21	802.15.3c

- Norme 802.11ad
  - Pour $R = 1/2$ $\rho(x) = \frac{1}{8}x^5 + \frac{3}{8}x^6 + \frac{3}{8}x^7 + \frac{1}{8}x^8$
  - Pour $R = 5/8$ $\rho(x) = \frac{1}{3}x^7 + \frac{1}{3}x^8 + \frac{1}{13}x^{10}$
  - Pour $R = 3/4$ $\rho(x) = \frac{1}{4}x^{13} + \frac{1}{2}x^{14} + \frac{1}{4}x^{15}$
  - Pour $R = 13/16$ $\rho(x) = \frac{1}{3}x^{14} + \frac{1}{3}x^{15} + \frac{1}{3}x^{16}$

- Norme 802.15.3c
  - Pour $R = 5/8$ $\rho(x) = \frac{1}{4}x^6 + \frac{1}{6}x^7 + \frac{1}{4}x^8 + \frac{1}{12}x^{11} + \frac{1}{12}x^{13} + \frac{1}{12}x^{14} + \frac{1}{12}x^{15}$
  - Pour $R = 3/4$ $\rho(x) = \frac{1}{4}x^{13} + \frac{1}{4}x^{14} + \frac{1}{4}x^{15} + \frac{1}{4}x^{16}$
  - Pour $R = 7/8$ $\rho(x) = \frac{1}{4}x^{29} + \frac{1}{4}x^{30} + \frac{1}{4}x^{31} + \frac{1}{4}x^{31}$

b. Norme 802.15.3c (deuxième structure)

N	K	R	Z	Normes
672	336	1/2	21	802.15.3c

Pour ce code $\rho(x) = \frac{1}{2}x^6 + \frac{1}{4}x^7 + \frac{1}{4}x^8$

c. Norme CCSDS 131.1(première structure)

N	K	R	Z	Normes
8 176	7 156	$\simeq 0.88$	511	CCSDS 131.1

Il s'agit d'un code LDPC régulier : $\rho(x) = x^{32}$.

d. Norme CCSDS 131.1 (deuxième structure)

N	K	R	Z	Normes
32 768	16 384	1/2	8192	CCSDS 131.1
24 576	16 384	2/3	4096	CCSDS 131.1
20 480	16 384	4/5	2048	CCSDS 131.1
8 192	4 096	1/2	2048	CCSDS 131.1
6 144	4 096	2/3	1024	CCSDS 131.1
5 120	4 096	4/5	512	CCSDS 131.1
2 048	1 024	1/2	512	CCSDS 131.1
1 536	1 024	2/3	256	CCSDS 131.1
1 280	1 024	4/5	128	CCSDS 131.1

Les polynômes indiquant la répartition de poids des lignes des matrices de parité sont :
— Pour $R = 1/2$ $\rho(x) = \frac{1}{3}x^3 + \frac{2}{3}x^6$
— Pour $R = 2/3$ $\rho(x) = \frac{1}{3}x^3 + \frac{2}{3}x^{10}$
— Pour $R = 3/4$ $\rho(x) = \frac{1}{3}x^3 + \frac{2}{3}x^{14}$
— Pour $R = 4/5$ $\rho(x) = \frac{1}{3}x^3 + \frac{2}{3}x^{18}$

e. Norme CCSDS 231.1

N	K	R	Z	Normes
512	256	1/2	8192	CCSDS 231.1
256	128	1/2	4096	CCSDS 231.1
128	64	1/2	2048	CCSDS 231.1

Quelle que soit la longueur considérée les lignes de la matrice de parité sont de poids 8 : $\rho(x) = x^8$.

A.2.2 Codes LDPC sans structure convolutive quasi-cycliques après permutation

a. Norme CMMB

N	K	R	Z	Norme
9 216	4 608	1/2	256	CMMB
9 216	6 912	3/4	256	CMMB

Ce sont des matrices régulières, les polynômes ρ sont donc des monômes :
— Pour $R = 1/2$ $\rho(x) = x^6$
— Pour $R = 3/4$ $\rho(x) = x^{12}$

b. Norme IEEE 802.15.3c (troisième structure)

N	K	R	Z	Norme
1 440	1 344	14/15	96	802.15.3c

Cette matrice de parité est régulière : $\rho(x) = x^{45}$.

A.2.3 Codes LDPC sans structure convolutive non quasi-cycliques

a. Normes IEEE 802.3 et 802.3an

N	K	R	Norme
2 048	1 723	$\simeq 0.84$	802.3

Cette matrice de parité est régulière : $\rho(x) = x^{32}$.

b. Norme IEEE 1901

Les longueurs des codes utilisés ne sont pas fixées, la matrice de parité étant infinie, il est simple de choisir la longueur souhaitée. La construction des matrices de parité est définie pour quatre rendements : 1/2, 2/3, 3/4 et 4/5.
Les polynômes définissant ces matrices de parité ont, pour un rendement donné, tous le même nombre de coefficients non-nuls. Les lignes de la matrice de parité infinie sont de poids :

— 6 pour $R = 1/2$
— 9 pour $R = 2/3$
— 12 pour $R = 3/4$
— 15 pour $R = 4/5$

Annexe B

Exemples de reconstruction de la permutation des turbo-codes

Pour ces exemples, les notations de la section 5.2 sont utilisées. Le code convolutif $\mathcal{C}_2$ est défini par $(1, \frac{1+D^2}{1+D+D^2})$, son treillis de codage est représenté sur la figure B.1 et la permutation π est de longueur $N = 8$.

B.1 Reconstruction de la permutation d'un turbo-code parallèle

Un ensemble de $M = 5$ mots sont observés en sortie d'un canal à effacement de probabilité $p = 0.10$. Ces mots bruités sont :

	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$	$\tilde{u}_4$	$\tilde{u}_5$	$\tilde{u}_6$	$\tilde{u}_7$	$\tilde{u}_8$	$\tilde{w}_1$	$\tilde{w}_2$	$\tilde{w}_3$	$\tilde{w}_4$	$\tilde{w}_5$	$\tilde{w}_6$	$\tilde{w}_7$	$\tilde{w}_8$
$\tilde{\boldsymbol{m}}^1$	1	0	1	0	1	1	1	1	1	0	1	1	?	0	1	1
$\tilde{\boldsymbol{m}}^2$	?	0	0	0	1	1	1	0	0	1	0	0	0	?	1	0
$\tilde{\boldsymbol{m}}^3$	1	1	0	?	0	1	0	1	0	?	1	0	1	0	1	1
$\tilde{\boldsymbol{m}}^4$	1	0	1	1	0	?	0	1	1	1	0	1	1	1	0	1
$\tilde{\boldsymbol{m}}^5$	0	0	0	0	0	1	1	1	?	0	0	0	1	?	0	1

Comme précédemment, le symbole ? représente l'observation d'un effacement. Afin de simplifier l'exemple, aucune probabilité n'est déterminée. Le canal considéré étant le canal à effacement, il suffit d'éliminer les indices impossibles puis de choisir un indice parmi ceux compatibles (ce choix s'effectue selon la proposition 5.3, mais l'exemple présenté ici est tel qu'un seul indice est compatible à chaque instant).

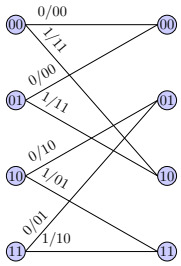


Figure B.1 – Treillis de codage du code $(1, \frac{1+D^2}{1+D+D^2})$

Recherche de $\pi(1)$

Par hypothèse, les registres sont initialisés à 00 au début du codage de chacun des mots. En utilisant le treillis de la figure B.1, il s'en déduit que $(u_{\pi(1)}, w_1) \in \{(0,0),(1,1)\}$.

Pour le mot $\tilde{\boldsymbol{m}}^1$, $\tilde{w}_1^1$ n'est pas un effacement, donc $w_1^1 = \tilde{w}_1^1 = 1$. D'où la nécessité que $u_{\pi(1)}^1 = 1$ et donc $\pi(1) \neq 2$ et $\pi(1) \neq 4$ puisque $\tilde{u}_2 = \tilde{u}_4 = 0$.
Le même raisonnement peut être effectué à partir du mot $\tilde{\boldsymbol{m}}^2$, pour celui-ci $\tilde{w}_1^2 = 0$ et donc $u_{\pi(1)}^2 = 0$, ce qui implique $\pi(1) \notin \{5,6,7\}$.
De même à partir des autres mots observés, le seul cas légèrement différent est celui du mot $\tilde{\boldsymbol{m}}^5$. En effet, pour celui-ci $\tilde{w}_1^5$ est un effacement, il est donc impossible d'en déduire les valeurs de w_1^5 et u_1^5. Ce mot n'apporte aucune information sur $\pi(1)$.

Le tableau ci-dessous regroupe les informations apportées par les cinq mots observés, le symbole $\surd$ représente une valeur possible (c'est-à-dire non éliminée) alors qu'un $\times$ indique une valeur impossible pour $\pi(1)$.

$\pi(1) =$	1	2	3	4	5	6	7	8
$\tilde{\boldsymbol{m}}^1$	$\surd$	$\times$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$
$\tilde{\boldsymbol{m}}^2$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\surd$
$\tilde{\boldsymbol{m}}^3$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\times$
$\tilde{\boldsymbol{m}}^4$	$\surd$	$\times$	$\surd$	$\surd$	$\times$	$\surd$	$\times$	$\surd$
$\tilde{\boldsymbol{m}}^5$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Il se déduit que $\pi(1) = 3$. En effet, c'est l'unique colonne ne contenant pas de $\times$, cet indice est donc compatible pour l'ensemble des mots observés.

État interne du codeur à l'instant $t = 1$

Pour chacun des mots l'état interne du codeur à l'instant $t = 0$ est 00, il faut maintenant déterminer cet état pour l'instant $t = 1$. Pour cela, il suffit d'utiliser le treillis. La connaissance de $\tilde{u}_{\pi(1)}$ ou $\tilde{w}_1$ suffit pour en déduire l'état recherché. D'où le tableau suivant :

état au temps $t =$	0	1
$\tilde{\boldsymbol{m}}^1$	00	10
$\tilde{\boldsymbol{m}}^2$	00	00
$\tilde{\boldsymbol{m}}^3$	00	00
$\tilde{\boldsymbol{m}}^4$	00	10
$\tilde{\boldsymbol{m}}^5$	00	00

Recherche de $\pi(2)$

De la même façon que pour la recherche de $\pi(1)$, il se déduit des valeurs de $\tilde{w}_2^1, \tilde{w}_2^2, \ldots, \tilde{w}_2^5$ un ensemble de valeurs impossibles pour $\pi(2)$. La première valeur impossible est 3 puisqu'elle est déjà choisie pour $\pi(1)$.

En utilisant le mot $\tilde{\boldsymbol{m}}^1$, on en déduit que $\pi(2) \notin \{2,4\}$. En effet, à l'instant $t = 1$, l'état interne du codeur pour ce mot est 10, en utilisant le treillis on en déduit que $(u_{\pi(2)}^1, w_2^1) \in$

$\{(0,1),(1,0)\}$. Or $\tilde{w}_2^1 = 0$ donc $u^1_{\pi(2)} = 1$.

En appliquant le même raisonnement sur l'ensemble des mots observés il découle le tableau suivant et le fait que $\pi(2) = 5$.

$\pi(2) =$	1	2	3	4	5	6	7	8
$\tilde{\boldsymbol{m}}^1$	$\surd$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$
$\tilde{\boldsymbol{m}}^2$	$\surd$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\times$
$\tilde{\boldsymbol{m}}^3$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$\tilde{\boldsymbol{m}}^4$	$\times$	$\surd$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\times$
$\tilde{\boldsymbol{m}}^5$	$\times$	$\surd$	$\times$	$\surd$	$\surd$	$\times$	$\times$	$\times$

État interne du codeur à l'instant $t = 2$

Grâce à la valeur de $\pi(2)$ on en déduit, pour l'ensemble des mots, l'état interne du codeur à l'instant $t = 2$:

état au temps $t =$	0	1	2
$\tilde{\boldsymbol{m}}^1$	00	10	01
$\tilde{\boldsymbol{m}}^2$	00	00	10
$\tilde{\boldsymbol{m}}^3$	00	00	00
$\tilde{\boldsymbol{m}}^4$	00	10	11
$\tilde{\boldsymbol{m}}^5$	00	00	00

Suite de la reconstruction

En continuant de la même manière, c'est-à-dire en déterminant la valeur de $\pi(t)$ puis en déterminant les états interne du codeur à l'instant t il en découle la permutation :

$$\pi \;=\; [3,5,1,2,8,6,4,7]$$

B.2 Reconstruction de la permutation d'un turbo-code parallèle poinçonné

Par hypothèse, le motif de poinçonnage utilisé est celui qui consiste à ne pas transmettre w_i lorsque i est impair. Un ensemble de $M = 5$ mots bruités sont observés en sortie d'un canal à effacement de probabilité $p = 0.10$:

	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$	$\tilde{u}_4$	$\tilde{u}_5$	$\tilde{u}_6$	$\tilde{u}_7$	$\tilde{u}_8$	$\tilde{w}_1$	$\tilde{w}_2$	$\tilde{w}_3$	$\tilde{w}_4$	$\tilde{w}_5$	$\tilde{w}_6$	$\tilde{w}_7$	$\tilde{w}_8$
$\tilde{\boldsymbol{m}}^1$	0	1	0	0	1	0	0	0		0		1		0		1
$\tilde{\boldsymbol{m}}^2$	1	0	0	1	1	1	e	0		0		0		e		0
$\tilde{\boldsymbol{m}}^3$	0	1	0	0	1	1	1	1		1		0		1		0
$\tilde{\boldsymbol{m}}^4$	0	1	1	1	0	1	0	1		0		1		0		e
$\tilde{\boldsymbol{m}}^5$	1	0	1	0	1	e	1	0		1		e		1		e

Dans cet exemple, les probabilités ne sont pas calculées, il suffit de trouver à chaque instant t quels sont les couples compatibles pour $(\pi(2t-1), \pi(2t))$.

$\pi(2)$ / $\pi(1)$	1	2	3	4	5	6	7	8
1	×	×			×			
2	×	×	×	×		×	×	×
3		×	×		×			
4		×		×	×			
5	×		×	×	×	×	×	×
6		×			×	×		
7		×			×		×	
8		×			×			×

TABLE B.1 – Recherche de $\pi(1)$ et $\pi(2)$.

$\pi(2)$ / $\pi(1)$	1	2	3	4	5	6	7	8
1	×	×	×	×	×	×	×	×
2	×	×	×	×	×	×	×	×
3	×	×	×	×	×	×	×	×
4	×	×	×	×	×		×	×
5	×	×	×	×	×	×	×	×
6	×	×	×		×	×	×	×
7	×	×	×	×	×	×	×	×
8	×	×	×	×	×	×	×	×

TABLE B.2 – Recherche de $\pi(1)$ et $\pi(2)$.

Recherche de $(\pi(1), \pi(2))$

Le plus simple pour trouver quels sont les couples compatibles pour $(\pi(1), \pi(2))$ est de remplir un tableau et d'éliminer au fur et à mesure les couples incompatibles. Par construction tous les couples de la forme (i, i) sont éliminés d'office.

Le codeur est initialisé à l'état 00 au début du codage de chaque mot, grâce à cette hypothèse et à partir du treillis de la figure B.1 on déduit que :
$(u_{\pi(1)}, u_{\pi(2)}, w_2) \in \{(0,0,0), (0,1,1), (1,1,0), (1,0,1)\}$.

En considérant uniquement le mot $\tilde{\boldsymbol{m}}^1$, comme $\tilde{w}_2^1 = 0$ il est impératif que :

$$(u^1_{\pi(1)}, u^1_{\pi(2)}) \in \{(0,0), (1,1)\}$$

Par conséquent tous les couples (i, j) tels que $\tilde{u}_i^1 \neq \tilde{u}_j^1$ sont éliminés. Ainsi le tableau obtenu en utilisant seulement le mot $\tilde{\boldsymbol{m}}^1$ est donné par la table B.1 (un × représente un couple incompatible).

La même démarche est ensuite effectuée à partir du mot $\tilde{\boldsymbol{m}}^2$. Il peut être remarqué que l'effacement de $\tilde{u}_7^2$ implique que tous les couples impliquant 7 sont conservés si seul ce mot bruité est considéré.

Finalement, en utilisant les cinq mots bruités, le tableau obtenu est celui de la table B.2, d'où $(\pi(1), \pi(2)) \in \{(4,6), (6,4)\}$.

État interne du codeur à l'instant 2

Avec le treillis de codage et les valeurs de $\tilde{u}_{\pi(1)}$, $\tilde{u}_{\pi(2)}$ et $\tilde{w}_2$ on déduit l'état interne du codeur à l'instant 2 pour chacun des mots observés. Ces états sont déterminés pour les deux couples de valeurs possibles pour $(\pi(1), \pi(2))$:

Si $(\pi(1), \pi(2)) = (4, 6)$:

état au temps $t =$	0	2
$\tilde{\boldsymbol{m}}^1$	00	00
$\tilde{\boldsymbol{m}}^2$	00	01
$\tilde{\boldsymbol{m}}^3$	00	10
$\tilde{\boldsymbol{m}}^4$	00	01
$\tilde{\boldsymbol{m}}^5$	00	10

Si $(\pi(1), \pi(2)) = (6, 4)$:

état au temps $t =$	0	2
$\tilde{\boldsymbol{m}}^1$	00	00
$\tilde{\boldsymbol{m}}^2$	00	01
$\tilde{\boldsymbol{m}}^3$	00	11
$\tilde{\boldsymbol{m}}^4$	00	01
$\tilde{\boldsymbol{m}}^5$	00	11

Recherche de $(\pi(3), \pi(4))$

Deux couples sont possibles pour $(\pi(1), \pi(2))$ il faut déterminer quel est le bon en recherchant les valeurs possibles pour $(\pi(3), \pi(4))$. Dans les deux cas, tous les couples impliquant un 4 ou un 6 sont incompatibles, ces valeurs étant déjà choisies.

Si $(\pi(1), \pi(2)) = (4, 6)$ alors en appliquant le même raisonnement que précédemment à partir des nouveaux états internes du codeurs tous les couples sont éliminés. Il n'y a aucune possibilité pour $(\pi(3), \pi(4))$, l'hypothèse $(\pi(1), \pi(2)) = (4, 6)$ est donc fausse d'où $(\pi(1), \pi(2)) = (6, 4)$.

En supposant que $(\pi(1), \pi(2)) = (6, 4)$, deux couples sont possibles pour $(\pi(3), \pi(4))$, il s'agit de $(2, 7)$ et $(7, 2)$.

Suite de la reconstruction

En continuant de la même façon le couple $(7, 2)$ est éliminé lors de la recherche de $(\pi(5), \pi(6))$. Finalement deux permutations sont compatibles avec les données reçues. Il est impossible d'en choisir l'une plus que l'autre, même en déterminant les probabilités exactes ou en ayant un nombre infini de mots observés. C'est deux permutations sont :

$$\pi = [6, 4, 2, 7, 3, 8, 1, 5] \text{ ou } \pi = [6, 4, 2, 7, 3, 8, 5, 1]$$

Annexe C

Turbo-codes parallèles poinçonnés : indéterminées lors de la reconstruction

Dans la sous-section 5.3.3 la possibilité que l'algorithme retourne plusieurs permutations ayant toutes la même probabilité d'être celle utilisée lors du codage a été évoquée. Cette annexe, classe les codes convolutifs $\mathcal{C}_2$ en fonction du nombre de permutations retournées lorsqu'ils sont utilisés. Une explication sera ensuite donnée à partir d'un exemple pour chacune des classes ainsi formées.

Généralement, les codeurs convolutifs utilisés sont composés, au maximum, de trois registres à décalages. C'est pour cette raison, que seuls ces codeurs sont considérés dans la suite de cette annexe. Les codes convolutifs sont, par hypothèse, de type $(2, 1)$ et le codage est systématique. Ces codes admettent donc une matrice génératrice de la forme $\boldsymbol{\mathcal{G}}(D) = (1 \quad P(D)/Q(D))$. Afin de simplifier les notations, seule la fraction $P(D)/Q(D)$ représentant la redondance sera inscrite. De plus, il est supposé, dans les hypothèses initiales, que le premier bit de redondance dépend du premier bit entré dans le codeur, ceci contraint $P(D)$ à avoir un coefficient de degré 0.

C.1 Nombre de permutations retournées

2 permutations

Lorsque les codes suivants sont utilisés, seulement deux permutations sont retournées par l'algorithme. La différence entre ces deux permutations porte uniquement sur les 2 derniers indices (ils sont inversés).

— $\frac{1}{1+D+D^2}$ — $\frac{1+D^2}{1+D+D^2}$ — $1+D+D^3$ — $\frac{1+D^2+D^3}{1+D}$

— $\frac{1}{1+D+D^3}$ — $\frac{1+D^2}{1+D+D^3}$ — $\frac{1+D+D^3}{1+D}$ — $\frac{1+D^2+D^3}{1+D^2}$

— $\frac{1}{1+D^2+D^3}$ — $\frac{1+D^2}{1+D^2+D^3}$ — $\frac{1+D+D^3}{1+D^2}$ — $\frac{1+D^2+D^3}{1+D+D^2+D^3}$

— $\frac{1+D}{1+D+D^3}$ — $1+D+D^2$ — $\frac{1+D+D^3}{1+D+D^2+D^3}$ — $\frac{1+D+D^2+D^3}{1+D+D^3}$

— $\frac{1+D}{1+D^2+D^3}$ — $\frac{1+D+D^2}{1+D^2}$ — $1+D^2+D^3$ — $\frac{1+D+D^2+D^3}{1+D^2+D^3}$

4 permutations

Si l'un des quatre codes suivants est utilisé, l'algorithme retourne 4 permutations. Les indéterminées portent sur les 4 derniers indices, $\pi(N)$ pouvant être inversé avec $\pi(N-1)$, et $\pi(N-2)$ peut être échangé avec $\pi(N-3)$.

— $\frac{1+D^3}{1+D+D^3}$ — $\frac{1+D^3}{1+D^2+D^3}$ — $\frac{1+D+D^3}{1+D^3}$ — $\frac{1+D^2+D^3}{1+D^3}$

8 permutations

Pour les deux codes suivants, l'algorithme retourne 8 permutations. Les valeurs de $\pi(N)$, $\pi(N-2)$ et $\pi(N-4)$ pouvant être échangées respectivement avec $\pi(N-1)$, $\pi(N-3)$ et $\pi(N-5)$.

— $\frac{1+D+D^3}{1+D^2+D^3}$ — $\frac{1+D^2+D^3}{1+D+D^3}$

$2^{\lceil \frac{N-3}{2} \rceil}$ permutations

Si le codeur $\mathcal{C}_2$ est l'un des codes ci-dessous, de très nombreuses permutations sont retournées. En effet, pour ces codes, il est impossible de déterminer si les valeurs de $\pi(t)$ et $\pi(t+3)$ ont été inversées ou non, pour toutes valeurs de t impaires.

— $\frac{1}{1+D^3}$ — $1+D^3$ — $\frac{1+D}{1+D+D^2}$ — $\frac{1+D+D^2}{1+D}$

$2^{\frac{N}{2}}$ permutations

Pour les quatre codes suivants, l'algorithme ne peut pas savoir si les valeurs de $\pi(2t)$ et $\pi(2t+1)$ ont été échangées ou non.

— $\frac{1}{1+D}$ — $1+D$ — $\frac{1}{1+D+D^2+D^3}$ — $1+D+D^2+D^3$

$\left(\frac{N}{2}\right)!$ permutations

La pire des situations se produit pour les deux codes suivants. Ils ne sont pas utilisés en pratique car ils se comportent comme deux codeurs indépendants. Pour ces codes, ils est impossible de savoir si $\pi(t)$ et $\pi(t+2k)$, $\forall t, \forall k$, ont été inversés lors de la reconstruction.

— $1+D^2$ — $\frac{1}{1+D^2}$

C.2 Justification du nombre de permutations retournées

Dans cette section, le contenu à l'instant t des trois registres du codeur convolutif sont notés a_t, b_t et c_t, l'état du codeur sera noté $e_t = (a_t, b_t, c_t)$. w_t représente la redondance émise

par le codeur à l'instant t et $x_t = u_{\pi(t)}$ correspond à l'entrée du codeur à cet instant.

2 permutations

Pour expliquer la situation, le code défini par $(1, \frac{1+D}{1+D^2+D^3})$ est pris pour exemple. Les valeurs des bits de redondance émis en fonction de l'état du codeur à l'instant t et des valeurs entrées jusque là sont déterminées :

$$\begin{aligned}
&w_t \text{ est poinçonné}\\
&w_{t+1} = x_{t+1} + x_t + a_t + c_t\\
&w_{t+2} \text{ est poinçonné}\\
&w_{t+3} = x_{t+3} + x_{t+2} + x_{t+1} + b_t\\
&w_{t+4} \text{ est poinçonné}\\
&w_{t+5} = x_{t+5} + x_{t+4} + x_{t+3} + x_t + b_t + c_t
\end{aligned}$$

D'après ces calculs, w_{t+1} fixe la valeur de $x_{t+1} \oplus x_t$, et non chacune des valeurs x_{t+1} et x_t. La redondance w_{t+3} ne dépend pas de x_t, c'est donc à cette étape que les valeurs respectives de x_t et x_{t+1} sont réellement fixées. Il y a désormais une indéterminée sur x_{t+2} et x_{t+3} bien que la valeur de $x_{t+2} \oplus x_{t+3}$ soit connue. Cette nouvelles indéterminée est levée grâce à w_{t+5} et ainsi de suite.
Finalement seule l'indéterminée sur les deux derniers indices ne peut être levée, c'est pour cette raison que l'algorithme retourne deux permutations $\tilde{\pi}_1$ et $\tilde{\pi}_2$ telles que :
$\forall t \in \{1, \ldots, N-2\}, \tilde{\pi}_1(t) = \tilde{\pi}_2(t)$, $\tilde{\pi}_1(N-1) = \tilde{\pi}_2(N)$ et $\tilde{\pi}_1(N) = \tilde{\pi}_2(N-1)$.

4 permutations

Pour cet exemple, le code considéré est défini par $(1, \frac{1+D^3}{1+D+D^3})$, les différentes redondances sont alors :

$$\begin{aligned}
&w_t \text{ est poinçonné}\\
&w_{t+1} = x_{t+1} + x_t + a_t + c_t\\
&w_{t+2} \text{ est poinçonné}\\
&w_{t+3} = x_{t+3} + x_{t+2} + x_{t+1} + x_t + b_t + c_t\\
&w_{t+4} \text{ est poinçonné}\\
&w_{t+5} = x_{t+5} + x_{t+4} + x_{t+3} + x_{t+2} + x_t + c_t\\
&w_{t+6} \text{ est poinçonné}\\
&w_{t+7} = x_{t+7} + x_{t+6} + x_{t+5} + x_{t+4} + x_{t+2} + a_t
\end{aligned}$$

w_{t+1} fixe la valeur de $x_{t+1} \oplus x_t$, la redondance w_{t+3} ne permet pas de lever cette indéterminée puisque x_t et x_{t+1} interviennent tous deux dans l'expression de w_{t+3}. C'est seulement avec la valeur de w_{t+5} que x_t et x_{t+1} sont fixées, c'est donc à cet instant que $\tilde{\pi}(t)$ et $\tilde{\pi}(t+1)$ sont déterminés. De même pour le couple $(\tilde{\pi}(t+2), \tilde{\pi}(t+3))$, il est déterminé grâce à la redondance émise à l'instant $t+7$ et ainsi de suite.

Les valeurs de $\tilde{\pi}(t)$ et $\tilde{\pi}(t+1)$ sont donc déterminées à l'instant $t+5$, il en découle l'impossibilité de résoudre les deux indéterminées sur les 4 derniers indices de la permutations.

8 permutations

La situation est similaire au cas précédent, sauf que cette fois l'indéterminée sur $\tilde{\pi}(t)$ et $\tilde{\pi}(t+1)$ est levée au temps $t+7$. L'algorithme retourne 8 permutations, les différences entre celles-ci portant uniquement sur les 6 derniers indices.

$2^{\lceil \frac{N-3}{2} \rceil}$ permutations

Dans ce cas, les indéterminées ne portent pas uniquement sur les derniers indices de la permutation. Comme illustré avec l'exemple suivant, $\pi(t)$ peut être inversé avec $\pi(t+3)$ pour toutes les valeurs de t impaires.
Pour comprendre la situation, il est nécessaire de calculer les redondances émises mais également l'évolution des états du codeur. Par exemple, avec le code défini par $(1, \frac{1}{1+D^3})$:

$$
\begin{aligned}
e_t &= (a_t, b_t, c_t) \\
&\qquad\qquad\qquad\qquad w_t \text{ est poinçonné} \\
e_{t+1} &= (x_t + c_t, a_t, b_t) \\
&\qquad\qquad\qquad\qquad w_{t+1} = x_{t+1} + b_t \\
e_{t+2} &= (x_{t+1} + b_t, x_t + c_t, a_t) \\
&\qquad\qquad\qquad\qquad w_{t+2} \text{ est poinçonné} \\
e_{t+3} &= (x_{t+2} + a_t, x_{t+1} + b_t, x_t + c_t) \\
&\qquad\qquad\qquad\qquad w_{t+3} = x_{t+3} + x_t + c_t \\
e_{t+4} &= (x_{t+3} + x_t + c_t, x_{t+2} + a_t, x_{t+1} + b_t) \\
&\qquad\qquad\qquad\qquad w_{t+4} \text{ est poinçonné} \\
e_{t+5} &= (x_{t+4} + x_{t+1} + b_t, x_{t+3} + x_t + c_t, x_{t+2} + a_t) \\
&\qquad\qquad\qquad\qquad w_{t+5} = x_{t+5} + x_{t+2} + b_t
\end{aligned}
$$

Avec les états et les redondances ci-dessus, il peut être remarqué que si les valeurs de x_t et x_{t+3} sont inversées alors les redondances émises sont les mêmes, puisque ces deux valeurs interviennent toujours simultanément dans les expressions des redondances. De plus, même s'il y a eu une inversion entre ces deux valeurs, l'état du codeur à l'instant $t+4$ est le même que sans cette inversion, les redondances et les états suivants sont donc eux aussi inchangés. Autrement dit, avec les codeurs de cet ensemble, les redondances émises sont les mêmes pour les deux mots d'information : $...x_t x_{t+1} x_{t+2} x_{t+3}...$ et $...x_{t+3} x_{t+1} x_{t+2} x_t....$ C'est pour cette raison que l'algorithme ne peut pas lever l'indéterminée portant sur $\pi(t)$ et $\pi(t+3)$, pour t impair.

Si la longueur de la permutation est paire, seules les valeurs de $\pi(2)$ et $\pi(N-1)$ sont fixées, alors que si la longueur est impaire, trois indices sont fixés : $\pi(2)$, $\pi(N-2)$ et $\pi(N)$, d'où le nombre de permutations retournées par l'algorithme $2^{\lceil \frac{N-3}{2} \rceil}$.

$2^{\frac{N}{2}}$ permutations

La situation est semblable à la précédente, soit le code défini par $(1, \frac{1}{1+D})$, les états du codeur et les redondances émises sont :

$$
\begin{aligned}
e_t &= (a_t) \\
&\qquad\qquad\qquad\qquad w_t \text{ est poinçonné} \\
e_{t+1} &= (x_t + a_t) \\
&\qquad\qquad\qquad\qquad w_{t+1} = x_{t+1} + x_t + a_t \\
e_{t+2} &= (x_{t+1} + x_t + a_t) \\
&\qquad\qquad\qquad\qquad w_{t+2} \text{ est poinçonné} \\
e_{t+3} &= (x_{t+2} + x_{t+1} + x_t + a_t) \\
&\qquad\qquad\qquad\qquad w_{t+3} = x_{t+3} + x_{t+2} + x_{t+1} + x_t + a_t \\
e_{t+4} &= (x_{t+3} + x_{t+2} + x_{t+1} + x_t + a_t)
\end{aligned}
$$

Dans ce cas, comme pour les autres codes de cet ensemble, les entrées x_t et x_{t+1}, pour t impair, peuvent être inversées sans répercussion sur les redondances émises. De plus, le codeur se retrouve à l'instant $t+4$ dans le même état que sans l'inversion, les redondances émises par la suite sont donc identiques. C'est pour cette raison que l'algorithme retourne $2^{\frac{N}{2}}$ permutations ayant toutes la même probabilité.

$\left(\frac{N}{2}\right)!$ permutations

Pour illustrer cette situation, le code $(1, \frac{1}{1+D^2})$ est utilisé. En effectuant le même raisonnement que pour les cas précédents, il se remarque rapidement que seuls les bits entrés en position paires sont présents dans les redondances non poinçonnées. Les registres à décalages correspondent à des accumulateurs, l'un sur les entrées paires l'autre sur les entrées impaires, c'est pour cette raison que tous les bits entrés en position impair peuvent être permutés entre eux sans influencer les redondances émises par le codeur. Les registres du codeur retrouvent finalement l'état qu'ils aurait du atteindre sans cette permutation. C'est à cause de cette configuration que l'algorithme retourne un nombre important de permutations ayant toutes la même probabilité.

Table des figures

Liste des tableaux

Liste des Algorithmes

Bibliographie

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. *IEEE Transactions on Information Theory*, 20(2) :284–287, 1974.

[2] J. Barbier. Reconstruction of turbo-code encoders. In *Proceedings of SPIE*, volume 5819, pages 463–473, 2005.

[3] J. Barbier and E. Filiol. Overview of Turbo-Code Reconstruction Techniques. *IACR Cryptology ePrint Archive*, 2009 :68, 2009.

[4] J. Barbier and J. Letessier. Forward Error Correcting Codes Characterization Based on Rank Properties. In *International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, November 2009. IEEE.

[5] J. Barbier, G. Sicot, and S. Houcke. Algebraic Approach of the Reconstruction of Linear and Convolutional Error Correcting Codes. In *World Academy of Science, Engineering and Technology*, volume 16, pages 66–71, November 2006.

[6] L. Bazzi, M. Mahdian, and D. Spielman. The minimum distance of turbo-like codes. *IEEE Transactions on Information Theory*, 55, January 2009.

[7] A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^{n/20}$: How 1+1=0 improves information set decoding. In *Advances in Cryptology, Eurocrypt 2012*. Springer, LNCS, 2012.

[8] M. Bellard. *Influence du mapping sur la reconnaissance d'un système de communication*. PhD thesis, Université Pierre et Marie Curie, Paris 6, January 2014.

[9] M. Bellard and J.-P. Tillich. Detecting and reconstructing an unknown convolutional code by counting collisions. In *International Symposium Information Theory (ISIT)*, pages 2967–2971, Honolulu, Hawaii, USA, July 2014. IEEE.

[10] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding : Turbo-codes. 1. In *International Conference on Communications (ICC), Technical program, Conference Record*, volume 2, pages 1064–1070, Geneva, Switzerland, May 1993. IEEE.

[11] P.L. Boyd and C. Robertson. Recovery of Unknown Constraint Length and Generator Polynomials for Linear Convolutional Encoders. In *21th Century Military Communications Conference*, volume 2, pages 947–951, Los Angeles, CA, USA, October 2000. IEEE.

[12] P.L. Boyd and C. Robertson. Recovery of Unknown Constraint Length and Generator Polynomials for Linear Convolutional Encoders in Noise. In *21th Century Military Communications Conference*, volume 2, pages 952–956, Los Angeles, CA, USA, October 2000. IEEE.

[13] G. Burel and R. Gautier. Blind Estimation of Encoder and Interleaver Characteristics in a Non Cooperative Context. In *IASTED International Conference on Communications, Internet and Information Technology (CIIT)*, Scottsdale, AZ, USA, November 2003.

[14] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code : Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *Transactions on Information Theory*, 44(1) :367–378, January 1998.

[15] CCSDS 131.1-O-2. Low Density Parity Check Codes for use in Near-Earth and Deep Space Applications, Orange Book. September 2007.

[16] CCSDS 131.5-O-1. Erasure Correcting Codes for use in Near-Earth and Deep-Space Communications, Orange Book. November 2014.

[17] CCSDS 231.1-O-1. Short Block Length LDPC Codes for TC synchronization and Channel Coding, Orange Book. April 2015.

[18] C. Chabot. Recognition of a code in a noisy environment. In *International Symposium Information Theory (ISIT)*, pages 2211–2215, Nice, France, June 2007. IEEE.

[19] C. Chabot. *Reconnaissance de codes, structure des codes quasi-cycliques.* PhD thesis, Université de Limoges, September 2009.

[20] W. Chen and G. Wu. Blind Recognition of (n-1)/n Rate Punctured Convolutional Encoders in a Noisy Environment. *Journal of Communications*, 10(4), April 2015.

[21] M. Cluzeau. Block code reconstruction using iterative decoding techniques. In *International Symposium Information Theory (ISIT)*, pages 2269–2273, Seattle, USA, July 2006. IEEE.

[22] M. Cluzeau. *Reconnaissance d'un schéma de codage.* PhD thesis, École Polytechnique, November 2006.

[23] M. Cluzeau and M. Finiasz. Reconstruction of Punctured Convolutional Codes. In *Information Theory Workshop (ITW)*, Taormina, Italy, October 2009. IEEE.

[24] M. Cluzeau and M. Finiasz. Recovering a Code's Length and Synchronization from a Noisy Intercepted Bitstream. In *International Symposium Information Theory (ISIT)*, pages 2737–2741, Seoul, Korea, 2009. IEEE.

[25] M. Cluzeau, M. Finiasz, and J.-P. Tillich. Methods for the Reconstruction of Parallel Turbo Codes. In *International Symposium Information Theory (ISIT)*, pages 2008–2012, Austin, Texas, USA, June 2010. IEEE.

[26] CMMB (China Multimedia Mobile Broadcasting) : Standard for Radio Film and Television Industry in P. R. China GY/T 220.1. Mobile Multimedia Broadcasting. Part 1 : Framing Structure, Channel Coding and Modulation for Broadcasting Channel. November 2006.

[27] M. Côte. *Reconnaissance de codes correcteurs d'erreurs.* PhD thesis, École Polytechnique, March 2010.

[28] M. Côte and N. Sendrier. Reconstruction of convolutional codes from noisy observation. In *International Symposium Information Theory (ISIT)*, pages 546–550, Seoul, Korea, 2009. IEEE.

[29] M. Côte and N. Sendrier. Reconstruction of a turbo-code interleaver from noisy observation. In *International Symposium Information Theory (ISIT)*, pages 2003–2007, Austin, Texas, USA, June 2010. IEEE.

[30] M.C. Davey and D.J.C MacKay. Low density parity check codes over GF(q). In *Information Theory Workshop (ITW)*, pages 70–71. IEEE, 1998.

[31] Z. Dawy, P. Hanus, J. Weindl, J. Dingel, and F. Morcos. On genomic coding theory. volume 18. Wiley, InterScience, April 2007.

[32] P.P. Debata, D. Mishra, K. Shaw, and S. Mishra. A Coding Theoretic Model for Error-detecting in DNA Sequences. *Procedia Engineering*, 38 :1773–1777, 2012.

[33] Y.G. Debessu, H.C. Wu, and H. Jiang. Novel Blind Encoder Parameter Estimation for Turbo Codes. *Communications Letters*, 16(12) :1917–1920, December 2012.

[34] Y.G. Debessu, H.C. Wu, H. Jiang, and S.Y. Chang. Blind Encoder Parameter Estimation for Turbo Codes. In *Globecom*, pages 4233–4237, Anaheim, USA, December 2012. IEEE.

[35] J. Dingel and J. Hagenauer. Parameter Estimation of a Convolutional Encoder from Noisy Observation. In *International Symposium Information Theory (ISIT)*, pages 1776–1780, Nice, France, June 2007. IEEE.

[36] D. Divsalar, S. Dolinar, and C. Jones. Low-rate LDPC codes with simple protograph structure. pages 1622–1626. IEEE, 2005.

[37] D. Divsalar, S. Dolinar, and C. Jones. Construction of Protograph LDPC Codes with Linear Minimum Distance. In *International Symposium Information Theory (ISIT)*, Seattle, USA, July 2006. IEEE.

[38] D. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin. A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 98–107. Springer, 2003.

[39] I. Dumer. On minimum distance decoding of linear codes. In *5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, Russia, 1991.

[40] ETSI EN 301 790 V1.5.1. Digital Video Broadcasting (DVB) ; Interaction channel for satellite distribution systems. May 2009.

[41] ETSI EN 302 307 part 2. Digital Video Broadcasting (DVB) ; Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Part II : S2-Extensions (DVB-S2X) - (Optional). DVB Document A83-2. March 2014.

[42] ETSI EN 302 307 V1.2.1. Digital Video Broadcasting (DVB) ; Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2). August 2009.

[43] ETSI EN 302 755 V1.3.1. Digital Video Broadcasting (DVB) ; Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2). April 2012.

[44] ETSI EN 302 755 V1.3.1 (final draft). Digital Video Broadcasting (DVB) ; Next Generation broadcasting system to Handheld, physical layer specification (DVB-NGH). November 2012.

[45] ETSI EN 302 769 V1.2.1. Digital Video Broadcasting (DVB) ; Frame structure channel coding and modulation for a second generation digital transmission system for cable systems (DVB-C2). April 2011.

[46] ETSI TS 101 376-5-3 V2.3.1. GEO-Mobile Radio Interface Specifications (Release 2) ; General Packet Radio Service ; Part 5 : Radio interface physical layer specifications ; Sub-part 3 : Channel Coding ; GMPRS-1 05.003. July 2007.

[47] ETSI TS 101 376-5-3 V3.1.1. GEO-Mobile Radio Interface Specifications (Release 3) Third Generation Satellite Packet Radio Service ; Part 5 : Radio interface physical layer specifications ; Sub part 3 : Channel Coding ; GMR-1 3G 45.003. July 2009.

[48] L.C.B. Faria, A.S.L Rocha, J.H. Kleinschmidt, R. Palazzo, and M.C. Silva-Filho. DNA sequences generated by BCH codes over GF(4). *Electronics Letters*, 46(3) :202–203, 2010.

[49] A. Jiménez Felström and K. Zigangirov. Time-Varing Periodic Convolutional Codes With Low Density Parity-Check Matrix. *IEEE Transactions on Information Therory*, 45(6) :2181–2191, September 1999.

[50] M. Ferrante and M. Saltalamacchia. The Coupon Collector's Problem. *MATerials MATemàtics*, 2014(2), May 2014.

[51] E. Filiol. Reconstruction of Convolutional Encoders over $GF(q)$. In *Cryptography and Coding : 6th IMA Int. Conf.*, pages 101–109, 1997.

[52] G.D. Forney. Performance and Complexity. *Shannon Lecture*, 1995.

[53] R. Gallager. Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, 8(1) :21–28, 1962.

[54] R. Gallager. A simple derivation of the coding theorem and some applications. *IEEE Transactions on Information Theory*, 11(1) :3–18, 1965.

[55] L. Gan, D. Li, Z.H. Liu, and L.P. Li. A low complexity algorithm of blind estimation of convolutional interleaver parameters. *Science China, Information Sciences*, 56(4), April 2013.

[56] J. Garcia-Frias and J.D. Villasenor. Combined Turbo Detection and Decoding for Unknown ISI Channels. *IEEE Transactions on Communications*, 51(1) :79–85, 2003.

[57] R. Gautier, G. Burel, M. Marazin, and C. Nsiala-Nzéza. Blind Estimation of Block Interleaver Length and Encoder Parameters. *MTA Review*, XXI(1) :31–44, March 2011.

[58] R. Gautier, M. Marazin, and G. Burel. Blind Recovery of the Second Convolutional Encoder of a Turbo-Code when its Systematic Outputs are Punctured. In *7-th IEEE-Communications 2008*, pages 345–348, Bucharest, Romania, 2008.

[59] IEEE Std 1901. IEEE Standard for Broadband over Power Line Networks : Medium Access Control and Physical Layer Specifications. December 2010.

[60] IEEE Std 202.20.2. IEEE Standard for Conformance to IEEE 802.20 System - Portocol Implementation Conformance Statement (PICS) Proforma. April 2010.

[61] IEEE Std 802.11. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. March 2012.

[62] IEEE Std 802.11ac. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4 : Enhancements for Very High Throughput for Operation in Bands below 6 GHz. December 2013.

[63] IEEE Std 802.11ad. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 3 : Enhancements for Very High Throughput in the 60 GHz Band. December 2012.

[64] IEEE Std 802.11af. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 5 : Television White Spaces (TVWS) Operation. December 2013.

[65] IEEE Std 802.11n. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 5 : Enhancements for Higher Throughput. October 2009.

[66] IEEE Std 802.15.3c. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 15.3 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs). Amendment 2 : Millimeter-wave-based Alternative Physical Layer Extension. October 2009.

[67] IEEE Std 802.16. IEEE Standard for Air Interface for Broadband Wireless Access Systems. August 2012.

[68] IEEE Std 802.16e. IEEE Standard for Local and metropolitan area networks. Part 16 : Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment 2 : Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands. February 2006.

[69] IEEE Std 802.20. IEEE Standard for Local and metropolitan area networks. Part 20 : Air Interface for Mobile Broadband Wireless Access Systems Supporting Vehicular Mobility—Physical and Media Access Control Layer Specification. August 2008.

[70] IEEE Std 802.22. Wireless Regional Area Networks (WRAN)—Specific requirements. Part 22 : Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications : Policies and Procedures for Operation in the TV Bands. July 2011.

[71] IEEE Std 802.3. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 3 : Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications . September 2008.

[72] IEEE Std 802.3an. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 3 : Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. Amendment 1 : Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T. September 2006.

[73] J. Jeong, D. Yoon, J. Lee, and S. Choi. Blind Reconstruction of a Helical Scan Interleaver. In *8th International Conference on Information Communications and Signal Processing (ICICS)*, Singapore, December 2011. IEEE.

[74] Y.-Q. Jia, L.-P. Li, Y.-Z. Li, and L. Gan. Blind estimation of communication emitter feature parameters. In *Proc. of International Conference on Computer and Information Technology (CIT)*, pages 281–285. IEEE, 2012.

[75] Y.Q. Jia, L.P. Li, Y.Z. Li, and L. Gan. Blind Estimation of Convolutional Interleaver Parameters. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Shanghai, China, September 2012. IEEE.

[76] I.S. Kang, H. Lee, S.J. Park, J.H. Soh, and Y.J. Song. Reconstruction Method for Reed-Muller Codes Using Fast Hadamard Transform. In *13th International Conference on Advanced Communication Technology (ICACT)*, pages 793–796, Seoul, Korea, February 2011.

[77] Y. Karimian and M.A. Attari. Recognition of channel encoder parameters from intercepted bitstream. In *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*, pages 1–5. IEEE, 2013.

[78] Y. Kou, S. Lin, and P.C. Fossorier. Low-Density Parity-Check Codes Based on Finite Geometries : A Rediscovery and New Results. *IEEE Transactions on Information Therory*, 47(7) :2711–2736, November 2001.

[79] S. Kudekar, T. Richardson, and R.L. Urbanke. Spatially coupled ensembles universally achieve capacity under Belief Propagation. *IEEE Transactions on Information Theory*, 59(12) :7761–7813, 2013.

[80] G. Landais. *Mise en oeuvre de cryptosystèmes basés sur les codes correcteurs d'erreurs et de leurs cryptanalyses.* PhD thesis, Université Pierre et Marie Curie, Paris 6, September 2014.

[81] G. Landais and J.P. Tillich. An efficient attack of a McEliece cryptosystem variant based on convolutional codes. In *Post-Quantum Cryptography*, pages 102–117. Springer, 2013.

[82] H. Lee, C.S. Park, J.H. Lee, and Y.J. Song. Reconstruction of BCH Codes Using Probability Compensation. In *18th Asia-Pacific Conference Communications (APCC)*, pages 591–594, Jeju Island, October 2012. IEEE.

[83] P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *C.G. Günther, Advances in Cryptology - EUROCRYPT '88*, volume 330, pages 275–280. Springer, 1988.

[84] J.S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5) :1354–1359, 1988.

[85] C. Li, T. Zhang, and Y. Liu. Blind recognition of RS codes based on Galois field columns Gaussian elimination. In *International Congress on Image and Signal Processing (CISP)*, pages 836–841. IEEE, 2014.

[86] D. Li and L. Guan. A Method of parameters estimation of SCCC Turbo code. In *Proc. of International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2013.

[87] W. Li, J. Lei, L. Wen, and B. Chen. An Improved Method of Blind Recognition of RS Code Based on Matrix Transformation. In *Proc. of ICCT (International Conference on Communication Technology)*. IEEE, November 2013.

[88] X. Liu and X. Geng. A convolutional code-based sequence analysis model and its application. *International journal of molecular sciences*, 14(4) :8393–8405, 2013.

[89] G. Liva, P. Pulini, and M. Chiani. On-Line Construction of Irregular Repeat Accumulate Codes for Packet Erasure Channels. *IEEE Transaction on Wireless Communications*, 12(2) :680–689, February 2013.

[90] C. Löndahl, T. Johansson, M. Koochak Shooshtari, M. Ahmadian-Attari, and M. Reza Aref. Squaring attacks on McEliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Designs, Codes and Cryptography*, 2015.

[91] L. Lu, K.H. Li, and Y.L. Guan. Blind Detection of Interleaver Parameters for Non-Binary Coded Data Streams. In *International Conference on Communications (ICC)*, Dresden, Germany, June 2009. IEEE.

[92] L. Lu, K.H. Li, and Y.L. Guan. Blind Identification of Convolutional Interleaver Parameters. In *7th International Conference on Information, Communications and Signal Processing (ICICS)*, Macau, China, December 2009. IEEE.

[93] O. Lu, L. Gan, and H. Liao. Blind Reconstruction of RS code. *Asian Journal of Applied Sciences*, (8) :34–45, 2015.

[94] P. Lu, S. Li, Y. Zou, and X. Luo. Blind recognition of punctured convolutional codes. *Science in China Series F : Information Sciences*, 48(4) :484–498, 2005.

[95] P. Lu, L. Shen, X. Luo, and Y. Zou. Blind Recognition of Punctured Convolutional Codes. In *International Symposium Information Theory (ISIT)*, page 457, Chicago, USA, July 2004. IEEE.

[96] P. Lu and Y. Zou. Fast Computations of Gröbner Bases and Blind Recognitions of Convolutional Codes. In C. Carlet and B. Sunar, editors, *First International Workshop Arithmetic of Finite Fields*, LNCS, pages 303–317, Madrid, Spain, June 2007. Springer.

[97] X. Luo, P. Wang, P. Lu, and F. Liu. Fast and Blind Restoration Scheme for the Initial Stats of LFSRs. In *5th International Multi-Symposiums on Computer and Computational Sciences (IMSCCS)*, pages 192–198. IEEE, 2006.

[98] D.J.C. MacKay and R.M. Neal. Near Shannon limit performance of Low Density Parity Check codes. *Electronics letters*, 32(18) :1645–1646, 1996.

[99] M. Marazin. *Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en oeuvre d'un récepteur intelligent.* PhD thesis, Université de Bretagne Occidentale, December 2009.

[100] M. Marazin, R. Gautier, and G. Burel. Blind Recovery of the Second Convolutional Encoder of a Turbo-Code when its Systematic Outputs are Punctured. *Military Technical Academy Review*, XIX(2) :213–232, June 2009.

[101] M. Marazin, R. Gautier, and G. Burel. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In *Globecom Workshops*, Honolulu, USA, 2009. IEEE.

[102] M. Marazin, R. Gautier, and G. Burel. Blind recovery of k/n rate convolutional encoders in a noisy environment. *EURASIP Journal on Wireless Communications and Neworking*, 2011.

[103] M. Marazin, R. Gautier, and G. Burel. Algebraic method for blind recovery of punctured convolutional encoders from an erroneous bitstream. *IET Signal Processing*, 6(2) :122–131, April 2012.

[104] M. Marazin, R. Gautier, and G. Burel. Some interesting dual-code properties of convolutional encoder for standards self-recognition. *Institution of Engineering and Technology Communications*, 6(8) :931–935, May 2012.

[105] A. May, M. Meurer, and E. Thomae. Decoding random linear codes in $\mathcal{O}(2^{0.054n})$. In *Asiacrypt 2011*, pages 107–124. Springer, LNCS, 2011.

[106] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. *PL DSN Progress Report*, pages 114–116, 1978.

[107] R. Moosavi and E.G. Larsson. A Fast Scheme for Blind Identification of Channel Codes. In *54th Globecom*, pages 1–5, Houston, TX, USA, December 2011. IEEE Press.

[108] A. Naseri, O. Azmoon, and S. Fazeli. Blind Recognition Algorithm of Turbo Codes for Communication Intelligence Systems. *International Journal of Computer Science Issues*, 8(1) :68–72, November 2011.

[109] C. O'Donoghue and C. Burkley. New algorithm to identify rate k/n catastrophic punctured convolutional encoders. In *Workshop on coding and cryptography (WCC)*, pages 405–412, Paris, France, 1999.

[110] E. Paolini, G. Liva, B. Matuz, and M. Chiani. Maximum Likelihood Erasure Decoding of LDPC Codes : Pivoting Algorithms and Code Design. *IEEE Transactions on Communications*, 60(11) :3209–3220, November 2012.

[111] E. Prange. The use of information sets in decoding cyclic codes. In *IRE Transactions, IT-8 : S5–S9*, 1962.

[112] Recommendation ITU-T G9960. Unified high-speed wireline-based home networking transceivers – System architecture and physical layer specification. December 2011.

[113] B. Rice. Determining the parameters of a rate $\frac{1}{n}$ convolutional encoder over $GF(q)$. In *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[114] G. L. Rosen. Examining coding structure and redundancy in DNA. *IEEE Engineering in Medecine and Biology*, 25(1) :62–68, January 2006.

[115] W. Ryan and S. Lin. *Channel Codes : Classical and Modern.* Cambridge University Press, 2009.

[116] H. Ryu, J. Lee, H. Hong, and D. Yoon. Estimation of interleaver period for unknown signals. In *Proceedings of International Conference on Network Infrastructure and Digital Content (IC-NIDC)*. IEEE, 2010.

[117] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27 :379–423 and 623–656, 1948.

[118] G. Sicot and S. Houcke. Blind detection of interleaver parameters. In *International Conference on Acoustics, Speech, and Signal Processing (ICCASP)*, volume 3, pages 829–832, Philadelphia, USA, March 2005. IEEE.

[119] G. Sicot and S. Houcke. Etude Statistique du seuil dans la detection d'entrelaceur. In *20ème Colloque sur le traitement du signal et des images*. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images, 2005.

[120] G. Sicot and S. Houcke. Theoretical study of the performance of a blind interleaver estimator. In *3rd International Symposium on Image/Video Communications over fixed and mobile networks (ISIVC)*, Hammamet, Tunisia, 2006.

[121] G. Sicot, S. Houcke, and J. Barbier. Blind Detection of Interleaver Parameters. *Signal Processing*, 89(4) :450–462, 2009.

[122] J. Stern. A method for finding codewords of small weight. In *Coding theory and applications*, pages 106–113. Springer, 1989.

[123] S. Su, J. Zhou, Z. Huang, C. Liu, and Y. Zhang. Blind identification of convolutional encoder parameters. *The Scientific World Journal*, 2014.

[124] F.W. Sun and A.J. Vinck. An Algorithm for Identifying Rate $(n-1)/n$ Catastrophic Punctured Convolutional Encoders. *IEEE Transactions on Information Theory*, 42(3) :1010–1013, May 1996.

[125] M. Teimouri and A. Hedayat. Parameter estimation of turbo code encoder. *Advances in Electrical Engineering*, 2014, 2014.

[126] J. Thorpe. Low-Density Parity-Check (LDPC) Codes Constructed from Protographs. *IPN Progress Report 42-154*, August 2003.

[127] J.-P. Tillich, A. Tixier, and N. Sendrier. Recovering the interleaver of an unknown turbo-code. In *International Symposium Information Theory (ISIT)*, pages 2784–2788, Honolulu, Hawaii, USA, July 2014. IEEE.

[128] A. Tixier. Blind identification of an unknown interleaved convolutional code. In *International Symposium Information Theory (ISIT)*, pages 71–75, Hong-Kong, China, June 2015. IEEE.

[129] A. Tixier. Blind identification of an unknown interleaved convolutional code. *arXiv.org, arXiv :1501.03715*, 2015.

[130] A. Valembois. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires.* PhD thesis, Université de Limoges, October 2000.

[131] A. Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111 :199–218, July 2001.

[132] A.J.H. Vinck, P. Dolezal, and Y.G. Kim. Convolutional Encoder State Estimation. *IEEE Transactions on Information Theory*, 44(4) :1604–1608, July 1998.

[133] A. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2) :260–269, 1967.

[134] F. Wang, Z. Huang, and Y. Zhou. A Method for Blind Recognition of Convolution Code Based on Euclidean Algorithm. In *International Conference on Wireless Communications and Mobile Computing*, pages 1414–1417, Shanghai, September 2007. IEEE.

[135] H. Wang, L. Yang, Y. Cai, and X. Luo. A Method for Blind Recognition of Low Code-rate Binary BCH Codes. In *International Conference on Multimedia Information Networking and Security (MINES)*. IEEE, November 2013.

[136] J. Wang, Y. Yue, and J. Yao. A Method of Blind Recognition of Cyclic Code Generator Polynomial. In *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. IEEE, 2010.

[137] J. Wang, Y. Yue, and J. Yao. Statistical Recognition Method of Binary BCH Code. *Communications and Network*, 3(1) :17–22, March 2011.

[138] L. Wang, Y. Hu, S. Hao, and L. Qi. The Method of Estimating the Length of Linear Cyclic Code Based on the Distribution of Code Weight. In *2nd International Conference on Information Science and Engineering (ICISE)*, pages 2459–2462. IEEE, 2010.

[139] L.X. Wang and D.F. Li. A New Method for BCH Codes of Blind Recognition. *Advanced Materials Research*, 631 :1403–1408, 2013.

[140] Y. Wang, F. Wang, and Z. Huang. Blind Recognition of (n, k, m) Convolutional Code Based on Local Decision in a Noisy Environment. In *International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE)*. Atlantis press, 2015.

[141] WiMedia Alliance. Multi Band OFDM Physical Layer Specification, PHY Specification : Final Delivrable 1.5. August 2009.

[142] T. Xia and H.-C. Wu. Joint blind frame synchronization and encoder identification for low-density parity-check codes. *IEEE Communications Letters*, 18(2), February 2014.

[143] T. Xia and H.C. Wu. Novel Blind Identification of LDPC Codes Using Average LLR of Syndrome a Posteriori Probability. In *12th International Conference on ITS Telecommunications*, pages 12–16, Taipei, China, 2012. IEEE.

[144] T. Xia and H.C. Wu. Blind Identification of Nonbinary LDPC Codes Using Average LLR of Syndrome a Posteriori Probability. *IEEE Communications Letters*, 17(7) :1301–1304, July 2013.

[145] T. Xia, H.C. Wu, S.Y. Chang, X. Liu, and S.Huang. Blind identification of binary LDPC codes for M-QAM signals. In *Global Communications Conference (GLOBECOM)*, pages 3532–3536. IEEE, 2014.

[146] H. Xie, X.M. Chai, F.H. Wang, and Z.T. Huang. A method for blind identification of rate 1/2 convolutional code based on improved Euclidean algorithm. In *11th International Conference on Signal Processing (ICSP)*, volume 2, pages 1307–1310, Beijing, China, October 2012. IEEE.

[147] H. Xie, F.H. Wang, and Z.T. Huang. Blind Recognition of Reed-Solomon Codes Based on Histogram Statistic of Galois Field Spectra. *Advanced Materials Research*, 791 :2088–2091, 2013.

[148] A.-D Yardi, S. Vijayakumaran, and A. Kumar. Blind reconstruction of binary cyclic codes. In *Proc of the European Wireless 2014*, 2014.

[149] A.D. Yardi and S. Vijayakumaran. Detecting Linear Block Codes in Noise using the GLRT. In *International Conference on Communications (ICC)*, pages 4895–4899, Budapest, Hungary, June 2013. IEEE.

[150] P. Yu, J. Li, and H. Peng. A Least Square Method for Parameter Estimation of RSC Sub-Codes of Turbo Codes. *IEEE Communications Letters*, 18(4), April 2014.

[151] P. Yu, J. Li, and H. Peng. Gibbs Sampling Based Parameter Estimation for RSC Sub-Codes of Turbo Codes. In *International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2014.

[152] H.L. Yudkin. *Channel state testing in information decoding.* PhD thesis, Dept. of Elec. Engg., M.I.T., 1964.

[153] H. Zhang, H.-C. Wu, and H. Jiang. Novel blind encoder identification of Reed-Solomon codes with low computational complexity. In *Global Communications Conference (Globcom)*, pages 3294–3299. IEEE, 2013.

[154] J. Zhou, Z. Huang, C. Liu, S. Su, and Y. Zhang. Information-Dispersion-Entropy-Based Blind Recognition of Binary BCH Codes in Soft Decision Situations. *Entropy*, 15(5) :1705–1725, 2013.

[155] J. Zhou, Z. Huang, S. Su, and S. Yang. Blind recognition of binary cyclic codes. *EURASIP Journal on Wireless Communications and Networking*, 2013.

[156] J. Zhou, Z. Huang, S. Su, and Y. Zhang. Blind identification of convolutional codes in soft-decision situations. *International Journal of Modern Communication Technologies and Research (IJMCTR)*, 2, April 2014.

[157] Y. Zrelli, R. Gautier, M. Marazin, E. Rannou, and E. Radoi. Focus on Theoretical Properties of Blind Convolutional Codes Identification Methods Based on Rank Criterion. In *9th International Conference on Communications*, pages 353–356, Bucharest, Romania, June 2012. IEEE.

[158] Y. Zrelli, R. Gautier, E. Rannou, M. Marazin, and E. Radoi. Blind identification of code word length for non-binary error-correcting codes in noisy transmission. *EURASIP Journal on Wireless Communications and Networking*, (1) :1–16, 2015.

[159] Y. Zrelli, M. Marazin, R. Gautier, and E. Rannou. Blind Identification of Convolutional Encoder Parameters over GF(2^m) in the Noiseless Case. In *20th International Conference on Computer Communications and Neteworks (ICCCN)*, Maui, HI, USA, 2011. IEEE.