Contribution à la construction et au décodage des codes polaires

Ludovic Chandesris

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE CERGY-PONTOISE

Spécialité : **Sciences et Technologies de l'Information et de la Communication (STIC)**

Présentée par

Ludovic Chandesris

Thèse dirigée par **Inbar Fijalkow** et codirigée par **Valentin Savin**

Préparée au sein de:
Laboratoire des Systèmes sans fils Haut Débit (LSHD, CEA-LETI)
Équipe de Traitement des Images et du Signal (ETIS, CNRS UMR 8051)

Contribution à la construction et au décodage des codes polaires

Version soumise en vue de la soutenance le 23 janvier 2019, devant le jury composé de :

Dr. Jean-Pierre TILLICH, HdR
Docteur-ingénieur de recherche INRIA, Le Chesnay, rapporteur

Prof. Jean-Claude BELFIORE
Professeur des Universités à Télécom-PariTech, Paris, rapporteur

Dr. Camille LEROUX
Maître de conférences à Bordeaux INP, Bordeaux, examinateur

Prof. Charly POULLIAT
Professeur des Universités à l'INP-ENSEEIHT, Toulouse, examinateur

Prof. David DECLERCQ
Professeur des Universités à l'ENSEA, Cergy-Pontoise, examinateur

Prof. Inbar FIJALKOW
Professeure des Universités à l'ENSEA, Cergy-Pontoise, directrice de thèse

Dr. Valentin SAVIN
Docteur-ingénieur de recherche CEA, encadrant

Remerciements

Avant de rentrer dans le vif du sujet, je tiens à remercier toutes les personnes qui m'ont accompagné tout au long de ces trois années et en ont fait une expérience humaine et intellectuelle extrêmement enrichissante. Toutes ont contribué, directement ou indirectement, à la qualité de ce manuscrit.

En tout premier lieu, je remercie chaudement Valentin Savin du CEA-Leti qui m'a encadré tout au long du parcours, pour sa bienveillance, son expertise, sa rigueur et sa pédagogie. Il a su trouver le bon équilibre entre me laisser explorer mes propres idées et réorienter les recherches lorsque je commencais à tourner en rond. Il a toujours pris le temps pour m'aider à assimiler les notions délicates rencontrées au fil la thèse, et j'espère que ce manuscrit fera honneur à son sens remarquable de la pédagogie. Un grand merci également à David Declercq, Professeur des Universités à l'ENSEA, qui, malgré la distance, a suivi de près l'évolution des recherches et a apporté sa grande expérience et ses précieux conseils sur bon nombre d'obstacles rencontrés. Malgré son emploi du temps chargé, il a systématiquement pris le temps de relire et de proposer des corrections sur les publications scientifiques et permis d'en améliorer significativement la qualité. Je remercie également Inbar Fijalkow, Professeure des Universités à l'ENSEA, qui a pris part à l'encadrement de la thèse, pour sa disponibilité et sa bienveillance.

Je tiens à exprimer ma gratitude envers mon jury de thèse, le président Charly Poulliat, Professeur des Universités à l'INP-ENSEEIHT, les rapporteurs Jean-Pierre Tillich, chercheur à l'INRIA, et Jean-Claude Belfiore, Professeur des Universités à TélécomParisTech, ainsi que les examinateurs Camille Leroux, Maître de conférences à Bordeaux INP, et David Declercq, Professeur des Universités à l'ENSEA, qui m'ont fait l'honneur de lire et de juger mon travail et qui ont su proposer des discussions très intéressantes.

J'adresse un grand merci aux équipes des laboratoires LSHD et LCOI du CEA-Leti, qui m'ont accueillies tout au long de ma thèse. A leur contact, j'ai découvert un environnement de travail aussi convivial et bienveillant qu'appliqué à contribuer à l'innovation scientifique et technologique. Je remercie les permanents du service Dimitri, Benoît M, Jean-Baptiste, Benoît D , François, Mickael, Antonio, Nicolas, Marc, Sébastien et les non-permanents Jéremy, Yoann M, Robin, Nicola, Imane, Tushar. Merci également aux anciens thésards Jimmy, Elodie, Yoann R, Gourab, Truong et Minh, pour toutes ces discussions, ces digresssions, ces suggestions et ces si précieux conseils pour la fin de ma propre thèse. Je n'oublie pas, bien sûr, mes collègues de bureau David, Patrick et Reda pour leur bienveillance, pour tous ces échanges, et en particulier pour avoir egayé mes quelques mois de rédaction. Finalement, je remercie chaleureusement Simon, pour tous ces moments partagés dans le bureau, dans un bar ou à la piscine, et Florian, pour son soutien indéfectible, notamment lors des derniers mois de thèse. Merci enfin à Sandrine, sécrétaire du laboratoire, pour l'organisation de la soutenance.

Finalement, je remercie mes proches, pour leur présence et leur soutien inconditionnel durant ces trois années, mes parents, mon frère Benoît, Ismène, Jean-Baptiste et Agathe.

Résumé

Dans un monde où les flux de données sont toujours plus intenses et omniprésents, les codes correcteurs d'erreurs constituent une composante essentielle aux systèmes de télécommunications, en permettant de compenser l'altération inévitable du message lors de la transmission, grâce à l'ajout de redondance. Dernière famille de codes à avoir été découverte, en 2008, les codes polaires ont rapidement attiré l'attention des chercheurs par leur aptitude à garantir une probabilité d'erreur arbitrairement faible pour tout rendement inférieur ou égal à la capacité du canal de propagation – la limite théorique, définie par Shannon en 1948. Tout récemment, ces codes ont été adoptés pour être utilisés dans la Cinquième génération de standards pour la téléphonie mobile 5G. Cette thèse porte sur l'étude de cette famille de codes face aux contraintes applicatives rencontrées en pratiques, et notamment dans la 5G, à travers les deux axes principaux que sont la construction des codes et leur décodage en longueur finie.

Après avoir passé en revue la définition et la caractérisation des codes polaires, ainsi que les aspects relatifs au spectre des distances, cette thèse explore la question de la flexibilité des codes en termes de longueurs, à l'aide des techniques de poinçonnage et de raccourcissement, et développe une méthodes pour optimiser le choix des motifs de sorte à minimiser le taux d'erreur pour le décodeur par annulation successive (SC, pour successive cancellation, en anglais). Dans un deuxième temps, un nouvel algorithme de décodage à inversion est introduit, permettant une amélioration significative des performances par rapport à l'état de l'art pour une complexité moyenne réduite et proche de celle du simple décodeur SC. Finalement, la question de l'utilisation des codes polaires avec les modulations d'ordre supérieur est abordée. Tout d'abord, les deux principales approches de modulations codées utilisant un code binaire, à savoir la modulation à entrelacement de bit et la modulation multi-niveaux, sont considérées, et il est mis en évidence le net avantage de la deuxième citée. Par suite, l'utilisation de codes polaires non-binaires est envisagée et il est montré qu'au prix d'une complexité de décodage supérieure, ceux-ci sont en mesure de surpasser leurs homologues binaires.

Abstract

In a world of intensive and widespread numeric dataflow, the use of error correcting codes has become an essential aspect of telecommunication systems, by allowing to compensate the unavoidable alteration of the messages during the transmission, by means of additional redundancy to the initial message. Last family of codes to be discovered, in 2008, polar codes have soon experienced a large interest of the community, due to their ability to achieve the Shannon capacity, namely the theoretical limit of the information rate, defined by Shannon in 1948. Very recently, these codes have been adopted in the fifth generation of cellular mobile communications 5G. This thesis concentrates on the analysis of this family of codes regarding constraints of practical systems, including in 5G, through the two major issues of the construction of the code and the decoding in finite length.

After reviewing the definition and the caraterization of polar codes and addressing the question of the spectrum distance, this thesis considers the issue of the flexibility of the codes as function of the codelength, using either puncturing or shortening techniques, and describes a new method to select the pattern so as to minimize the error rate under successive cancellation decoding (SC). Secondly, a new bit-flipping decoding algorithm is proposed, improving significantly the performance compared to state-of-the-art approach, while offering a reduced average complexity, which stays close to the one of the SC decoder. Thirdly, the issue of the use of polar codes with high order modulation is considered. On the one hand, the case of coded modulations, where a high order modulation is combined with binary codes, is considered through the bit-interleaved coded modulation and the multilevel coded modulation, and it is shown that the second one is very well suitable for polar codes. On the other hand, the case of non-binary codes, which can be mapped directly to the constellation, are considered and it is revealed they are able to surpass their binary counterpart at the cost of additional decoding complexity.

Publications

Brevets

- **"Method for decoding a polar code with inversion of unreliable bits"**
  "méthode de décodage à inversion des Codes Polaires"
  Numéro du brevet : WO2017/178567, *Brevet déposé en Octobre 2017*

Journaux internationaux

- **"Dynamic SCFlip Decoding for Polar Codes"**
  Ludovic Chandesris, Valentin Savin, David Declercq
  IEEE Transaction on Communications, Vol 66, 2018.

Conférences internationales

- **"An Improved SCFlip Decoder for Polar Codes"**
  Ludovic Chandesris, Valentin Savin, David Declercq
  IEEE Global Communications Conference (GLOBECOM), Washington, USA, 2016.
- **"On puncturing strategies for polar codes"**
  Ludovic Chandesris, Valentin Savin, David Declercq
  IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 2017.
- **"Lasting Successive-Cancellation based Decoders for Multilevel Polar Coded Modulation"**
  Ludovic Chandesris, Valentin Savin, David Declercq
  25th International Conference on Telecommunication, Saint-Malo, France, 2018.

Conférences nationales

- **"Un décodeur à inversion dynamique pour les codes polaires"**
  Ludovic Chandesris, Valentin Savin, David Declercq
  Colloque Gretsi, Juan-Les-Pins, France, Sept. 2017.

Table des Matières

Liste des Figures

Liste des Tableaux

Liste des Acronymes

Note : les abréviations utilisées correspondent aux termes anglais, car largement répandues au sein de la communauté.

5G	Fifth generation of cellular mobile communications	Cinquième génération de standards pour la téléphonie mobile
3GPP	3rd Generation Partnership Project	Troisème génération de projet partenaire
SNR	Signal-to-Noise Ratio	Rapport Signal à Bruit
BEC	Binary Erasure Channel	Canal à effacement
(BI-)AWGN	(Binary-Input) Additive White Gaussian Noise	Canal (binaire) avec bruit blanc gaussien additif
(B)-DMC	(Binary) Discrete memoryless channel	Canal (binaire) discret sans mémoire
UB	Union Bound	Borne de l'union
ML	Maximum likelihood	Maximum de vraisemblance
MS	Min-Sum	Minimum-sommation
DE	Density Evolution	Évolution de densité
DE-GA	Density Evolution by Gaussian Approximation	Évolution de densité par approximation gaussienne
LDPC	Low Density Parity Check codes	Codes à matrice de parité creuse
CMD	Decreasing monomial code	Code monomial décroissant
SDO	Spectrum distance optimum code	Code à spectre des distances optimal
RM	Reed-Muller code	Code Reed-Muller
SER	Symbol Error Rate	Taux d'erreur symbole
BER	Bit Error Rate	Taux d'erreur bit
WER	Word Error Rate	Taux d'erreur bloc
BICM	Bit-Interleaved Coded Modulation	Modulation codée à entrelacement des bits
MLCM	Multilevel Coded Modulation	Modulation codée multi-niveaux
SP	Set-Partitioning labelling	Attribution par partition d'ensemble
BPSK	Binary Phase Shift Keying	Modulation binaire par changement de phase
QAM	Quadrature Amplitude Modulation	Modulation d'amplitude en quadrature

BP	Belief Propagation	Propagation de croyance
QUP	Quasi-Uniform Puncturing	Poinçonnage quasi-uniforme
QUS	Quasi-Uniform Shortening	Raccourcissement quasi-uniforme
LLR	Log Likelihood Ratio	Rapport de vraisemblance logarithmique
CRC	Cyclic Redundancy Check	Contrôle de redondance cyclique
PDF	Probability Distribution Function	Densité de probabilité
SC	Successive Cancellation decoder	Décodeur par annulation successive
SSC	Simplified Successive Cancellation decoder	Décodeur par annulation successive simplifié
SCL	Successive Cancellation List decoder	Décodeur par liste à annulation successive
CA-SCL	CRC-aided SCL decoder	Décodeur SCL aidé par un CRC
SCS	Successive Cancellation Stack decoder	Décodeur de Stack à annulation successive
SCFlip	Successive Cancellation Flip decoder	Décodeur à inversion
DSCFlip	Dynamic Successive Cancellation Flip decoder	Décodeur dynamique à inversion

Introduction

Il y a exactement 70 ans, en 1948, C. Shannon publiait l'article fondateur de la théorie de l'information "The Mathematical Theory of Information" [90], formalisation mathématiquement les concepts fondamentaux relatifs à la transmission d'information. En particulier, il y introduit la notion de capacité d'un canal bruité, qui porte désormais son nom, comme le débit d'information théorique maximal en présence de bruit. Cette notion trouve sa pertinence dans son fameux "deuxième théorème", dit de codage canal, qui affirme qu'en ajoutant de la redondance au message initial, il est possible de transmettre avec une probabilité d'erreur arbitrairement faible, pour tout rendement inférieur à la capacité du canal. Cependant, aucune méthode explicite n'est fournie pour construire des codes capables d'atteindre, ni même d'approcher, cette limite théorique.

Dans les années qui suivirent, une large variété de codes correcteurs furent proposés afin de répondre à la problématique soulevée par Shannon, tels que les codes Reed-Muller [71], Reed-Solomon (RS) [79], les codes convolutifs [29] ou encore les codes LDPC (Low density parity check) [36]. Cependant, il faut attendre les années 1990 pour que le codage connaisse un second souffle, grâce à la fois à la découverte par C. Berrou en 1993 des codes Turbo [57], et à la résurgence des codes LDPC, suite aux travaux de McKay [61] en 1996, puis à ceux de Richardson et *al.* introduisant la technique d'évolution de densité pour l'analyse et l'optimisation de ces codes [25]. Toutefois, malgré ces progrès considérables, la question de l'atteignabilité de la limite de Shannon continuait de résister aux chercheurs. Il faudra attendre jusqu'en 2008, soit 60 après la formulation par Shannon de son fameux théorème, pour qu'Arikan apporte finalement la solution par les codes polaires. Si, depuis, il a été montré que les SC-LDPC (Spatially-Coupled LDPC) en font autant, les codes polaires sont apparus comme une solution particulièrement prometteuse du fait qu'un décodeur par annulation successive (SC, pour Successive Cancellation) de faible complexité est suffisant pour atteindre asymptotiquement la borne théorique. De plus, pour pallier aux performances limitées du SC en longueur finie, le décodage par liste [96], dit SC-Liste, proposé quelque temps plus tard, permettait aux codes polaires de concurrencer les performances des codes LDPC et Turbo. En une décennie, les codes polaires en sont venus à jouer les premiers rôles dans le domaine des codes correcteurs, entériné par leur récente adoption pour la cinquième génération de standards de téléphonie mobile 5G, aux côtés des incontournables LDPC.

Cette thèse se concentre sur l'étude des codes polaires et de leurs propriétés les rendant attractifs pour les applications pratiques, ainsi que sur la question du décodage efficace de ces codes, avec, en ligne de mire, les contextes applicatifs de la 5G. Deux axes principaux et complémentaires sont étudiés, à savoir la question de la construction des codes polaires ainsi que celle du décodage en longueur finie. La construction des codes polaires consiste à s'efforcer de sélectionner, parmi l'ensemble des codes de la famille des codes polaires, celui offrant les meilleures performances. Si des solutions efficaces existent lorsqu'un décodeur SC est utilisé, il s'avère qu'il n'en est pas de même pour les autres algorithmes de décodage. De plus, la structure des codes polaires, contraignant la longueur du code à une puissance de deux, soulève la question de l'emploi de techniques de poinçonnage ou de raccourcissement. Finalement, ces deux derniers

aspects se retrouvent également lorsqu'une modulation d'ordre supérieure est utilisée. Le deuxième axe de la thèse constitue le décodage des codes polaires en longueur finie. Notamment, la recherche d'un compromis intermédiaire entre le décodeur SC, de faible complexité mais aux performances insuffisantes en longueur finie, et le décodeur par liste, excellent en termes de performances, mais dont la complexité et la latence peuvent être un frein en pratique.

Organisation de la thèse

Le **premier chapitre** de la thèse revient sur la définition des codes polaires et sur le phénomène de polarisation à l'origine de leur aptitude à atteindre la capacité de Shannon. L'algorithme fondamental de décodage, appelé décodage par annulation successive, est détaillé, ainsi que les méthodes de l'état de l'art utilisées pour optimiser le code polaire pour ce décodeur. Les propriétés connues sur le spectre des distances des codes polaires sont également abordées, auxquelles s'ajoutent une contribution supplémentaire permettant de déterminer le code maximisant la distance et minimisant la multiplicité des mot de code de poids faible. Finalement, les principales alternatives au décodeur SC pour la longueur finie sont passées en revue, en particulier le décodeur par liste, le décodeur à inversion et le décodeur séquentiel.

Le **deuxième chapitre** explore la question de la flexibilité sur la longueur du code, indispensable dans les standards de télécommunications, à travers les techniques du poinçonnage et du raccourcissement des codes polaires. Il est montré que, du fait de la structure particulière des codes polaires, la recherche du meilleur motif pour le décodeur SC peut être réduite par rapport à une recherche exhaustive. Deux catégories de motifs sont définies et caractérisées, et un algorithme est proposé afin d'énumérer les motifs de chaque catégorie. Il est ensuite montré que les problématiques du poinçonnage et du raccourcissement sont étroitement liées, de sorte que les résultats obtenus dans le premier cas sont transposés au cas du raccourcissement. Dans un second temps, il est montré que les codes polaires poinçonnés ou raccourcis peuvent, sous certaines conditions, être assimilés à des codes polaires utilisant une combinaison de noyaux et vice-versa. Ce parallèle donnera lieu à une approche radicalement différente pour l'optimisation des motifs de poinçonnage, basée sur le critère de l'exposant d'erreur des matrices noyaux.

Le **troisième chapitre** décrit un nouvel algorithme de décodage pour les codes polaires, appelé décodeur à inversion dynamique. Partant du décodeur à inversion existant dans l'état de l'art, une analyse précise de ses limites est présentée, et des solutions concrètes pour les surmonter sont proposées, tant au niveau de la définition de la métrique, que du processus de sélection des nouvelles tentatives. Il est montré que le décodeur proposé est capable de se rapprocher des performances du décodeur par liste, tout en conservant une complexité moyenne similaire à celle du SC. Finalement, il est également mis en évidence le fait que la métrique proposée est susceptible d'améliorer les performances du décodeur séquentiel.

Le **quatrième chapitre** développe la question des modulations codées utilisant des codes polaires binaires, à travers la comparaison des deux principales approches que sont la modulation codée à entrelacement de bit (BICM pour Bit-Interleaved Coded Modulation en anglais) et la modulation codée multi-niveaux (MLCM pour Multilevel Coded Modulation en anglais). Il est notamment montré que la problématique centrale de la MLCM, à savoir le choix des rendements des codes sur chaque niveau binaire, peut être efficacement résolue pour un décodeur SC et une nouvelle méthode est proposée pour le cas du décodeur par liste. Finalement, il sera mis en évidence l'étroite compatibilité entre les codes polaires et la modulation MLCM, permettant l'optimisation du décodage multi-niveaux, et offrant un avantage significatif par rapport à la modulation BICM.

Le **cinquième chapitre** s'intéresse à une question encore assez peu explorée pour les codes polaires, à savoir le cas des codes polaires non-binaires. Plus précisément, une généralisation de la matrice noyau d'Arikan sur un alphabet non-binaire est considérée, en intégrant (au noyau) une permutation de l'alphabet non-binaire. Trois cas de figures sont successivement discutés, suivant que les permutations de l'alphabet non-binaire sont définies dans le groupe symétrique, dans le groupe général linéaire, ou dans le corps de Galois. Une méthode est proposée afin d'optimiser les permutations dans chacun de ses trois ensembles. Finalement, il est mis en évidence qu'un gain significatif peut être obtenu en optimisation spécifiquement les permutations, et que les codes non-binaires obtenus supplantent significativement leurs homologues binaires aussi bien sur un canal binaire que non-binaire.

1 Généralités sur les codes polaires

Contenu du Chapitre

Ce chapitre introduit le phénomène de polarisation et la définition des codes polaires, d'après l'article de référence d'Arikan [5]. Il revient également sur certaines propriétés de la famille des codes polaires, la question de la construction des codes polaires, ainsi que sur les principaux algorithmes de décodage. Certaines contributions originales ont été insérées dans ce chapitre afin de les présenter dans leur contexte, mais la distinction avec l'état de l'art sera clairement indiquée.

1.1 Phénomène de polarisation

1.1.1 Canal binaire discret sans mémoire

Soit W un canal binaire discret sans mémoire (B-DMC pour binary-input discret memoryless channel, en anglais), *i.e.*:

- L'alphabet d'entrée $\mathcal{X}$ est un ensemble à deux éléments $\mathcal{X} = \{0,1\}$
- L'alphabet de sortie $\mathcal{Y}$ est fini
- Pour tout $(x_0,\dots,x_{N-1}) \in \mathcal{X}^N$ transmis et $(y_0,\dots,y_{N-1}) \in \mathcal{Y}^N$ reçu, la propriété suivante est vérifiée:

$$W^N(y_0,\dots,y_{N-1}\,|\,x_0,\dots,x_{N-1}) = W(y_0\,|\,x_0)\cdots W(y_{N-1}\,|\,x_{N-1}), \tag{1.1}$$

  où :

  - $W(y_i\,|\,x_i)$ correspond à la probabilité de recevoir y_i lorsque x_i est transmis
  - W^N correspond au canal vectoriel correspondant à N utilisations successives de W

On supposera de plus que W est symétrique, *i.e.* il existe une permutation $\pi : \mathcal{Y} \to \mathcal{Y}$, telle que $\pi = \pi^{-1}$ et $W(y|1) = W(\pi(y)|0), \forall y \in \mathcal{Y}$, et cette hypothèse sera faite implicitement tout au long de ce chapitre. La capacité symétrique[1] du canal W est définie par la quantité :

$$I(W) = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \frac{1}{2} W(y|x) \log_2 \left(\frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \right), \tag{1.2}$$

et le coefficient de Bhattacharyya par:

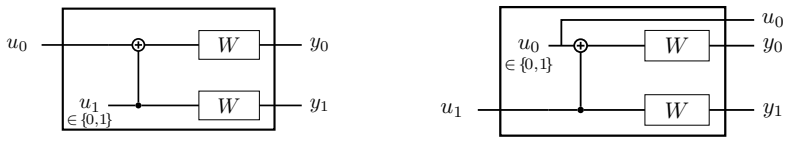
$$Z(W) = \sum_{y\in\mathcal{Y}} \sqrt{W(y|0)W(y|1)} \tag{1.3}$$

Ces deux grandeurs sont toutes deux dans l'intervalle $[0,1]$. La capacité symétrique d'un canal W correspond, d'après le second théorème de Shannon, au rendement théorique maximal auquel il est possible de transmettre avec une probabilité d'erreur arbitrairement faible. Le paramètre de Bhattacharyya, quant à lui, permet de quantifier la fiabilité du canal W. Ces deux paramètres sont liés par la double inégalité suivante :

$$\log_2 \left(\frac{2}{1+Z(W)} \right) \leq I(W) \leq \sqrt{1 - Z(W)^2} \tag{1.4}$$

En particulier, $I(W) = 1 \iff Z(W) = 0$ et $I(W) = 0 \iff Z(W) = 1$.

[1] Notons que, pour un canal à entrée binaire, la capacité d'un canal symétrique est égale à sa capacité symétrique.



(a) Canal virtuel W^- (b) Canal virtuel W^+

FIGURE 1.1: *Représentation des canaux virtuels W_- (a) et W_+ (b)*

1.1.2 TRANSFORMATION ÉLÉMENTAIRE

Soit la matrice G_2, appelée matrice noyau, définie par :

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{1.5}$$

Soit $u_0^1 \in \mathcal{X}^2$. La séquence codée à transmettre sur le canal W est donnée par $x_0^1 = u_0^1 \cdot G_2 = (u_0 + u_1, u_1)$. On définit le canal vectoriel, noté W_2, d'entrée $(u_0, u_1) \in \mathcal{X}^2$ et de sortie $(y_0, y_1) \in \mathcal{Y}^2$, et de probabilité de transition :

$$W_2(y_0, y_1 | u_0, u_1) = W(y_0 | u_0 + u_1) \cdot W(y_1 | u_1) \tag{1.6}$$

Ce canal vectoriel est alors divisé en deux canaux virtuels W^+ et W^-, caractérisés par :

$$W^- : \begin{array}{l} \mathcal{X} \mapsto \mathcal{Y}^2 \\ u_0 \to (y_0, y_1) \end{array} \qquad W^+ : \begin{array}{l} \mathcal{X} \mapsto \mathcal{Y}^2 \times \mathcal{X} \\ u_1 \to (y_0, y_1, u_0) \end{array}$$

avec pour probabilités de transition:

$$\begin{aligned} W^-(y_0, y_1 | u_0) &= \frac{1}{2} \sum_{u_1} W_2(y_0, y_1 | u_0, u_1) \\ W^+(y_0, y_1, u_0 | u_1) &= \frac{1}{2} W_2(y_0, y_1 | u_0, u_1) \end{aligned}$$

L'interprétation de ces canaux virtuels est donnée dans la figure 1.1. Le canal virtuel W^- tire la valeur de u_1 aléatoirement (avec $\Pr(u_1 = 0) = \Pr(u_1 = 1) = \frac{1}{2}$) et transmet $(u_0 + u_1, u_1)$ sur le canal W. Le canal virtuel W^+ tire la valeur de u_0 aléatoirement, transmet $(u_0 + u_1, u_1)$ sur le canal W, mais fourni également la valeur de u_0. Il peut être montré que [5]:

$$I(W^+) + I(W^-) = 2I(W) \tag{1.7}$$

$$I(W^-) \leq I(W^+) \tag{1.8}$$

avec égalité si et seulement si $I(W) \in \{0,1\}$. Un cas très particulier, mais qui servira à titre illustratif, est le cas du canal à effacement (BEC pour Binary Erasure Channel en anglais), pour lequel les canaux virtuels W^- et W^+ sont eux-mêmes assimilables à des canaux BEC dont les capacités symétriques sont données par [5] :

$$I(W^-) = I(W)^2 \tag{1.9}$$

$$I(W^+) = 2I(W) - I(W)^2 \tag{1.10}$$

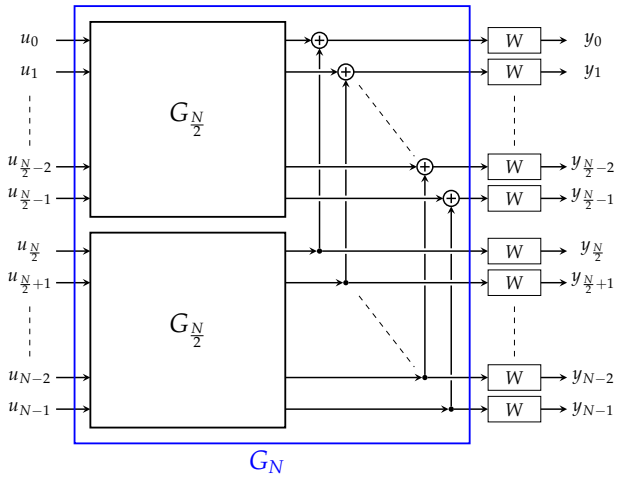


FIGURE 1.2: *Construction récursive du graphe*

A noter que pour un canal BEC, l'information mutuelle et le coefficient de Bhattacharrya sont liés par la relation $I(W) + Z(W) = 1$.

On a donc, à partir de deux réalisations indépendantes du même canal B-DMC W et en appliquant la transformation linéaire décrite par la matrice G_2, défini un canal vectoriel W_2 (opération appelée "channel combining" par Arikan), puis subdivisé celui-ci en deux canaux (virtuels) B-DMC W^+ et W^- (opération appelée "channel splitting" par Arikan), dont l'un est amélioré et l'autre détérioré par rapport au canal W. Dans la suite, on s'intéressera à la transformation $(W, W) \mapsto (W^-, W^+)$ permettant de passer des deux réalisations indépendantes de W aux deux canaux virtuels.

1.1.3 CONSTRUCTION RÉCURSIVE

Le principe est d'exploiter la transformation $(W, W) \mapsto (W^-, W^+)$ de manière récursive afin de générer, à partir de N réalisations indépendantes du même canal W, N canaux virtuels dont les capacités symétriques sont progressivement *polarisées* au fur et à mesure des étapes de la récurrence, c'est-à-dire que leurs informations mutuelles tendent asymptotiquement vers 0 ou 1.

La transformation linéaire à appliquer sur la séquence u_0^{N-1} de manière à générer la polarisation des canaux virtuels est décrite par la matrice $G_N = G_2^{\otimes n}$ définie comme la n-ième puissance de Kronecker de la matrice G_2, et où $N = 2^n$. La transformation récursive s'écrit donc:

$$\begin{aligned} x_0^{N-1} &= u_0^{N-1} \cdot G_N \\ &= u_0^{N-1} \cdot (G_{N/2} \otimes G_2) \\ &= (u_0^{N/2-1} \cdot G_{N/2} + u_{N/2}^{N-1} \cdot G_{N/2} \ , \ u_{N/2}^{N-1} \cdot G_{N/2}) \end{aligned}$$

Cette récurrence est visualisée sur la figure 1.2. Similairement à ce qui a été fait au niveau du noyau, il est possible, partant de $N = 2^n$ réalisations indépendantes du même canal W, de définir un canal vectoriel $W_N : \mathcal{X}^N \mapsto \mathcal{Y}^N$, puis de le sub-diviser en N canaux virtuels $W_N^{(i)} : \mathcal{X} \mapsto \mathcal{Y}^N \times \mathcal{X}^{i-1}$, d'entrée u_i et de sortie (y_0^{N-1}, u_0^{i-1}), et de probabilité de transition:

$$W_N^{(i)}(y_0^{N-1}, u_0^{i-1} \,|\, u_i) = \sum_{u_{i+1}^N \in \mathcal{X}^{N-i}} \frac{1}{2^{N-1}} W_N(y_0^{N-1} \,|\, u_0^{N-1}) \tag{1.11}$$

La capacité symétrique d'un canal virtuel est notée $I(W_N^{(i)})$, et il s'agit de s'intéresser à la distribution des valeurs $I(W_N^{(i)})$ pour tout $i \in \{0, \ldots, N-1\}$ lorsque N tend vers l'infini. Une illustration de l'évolution

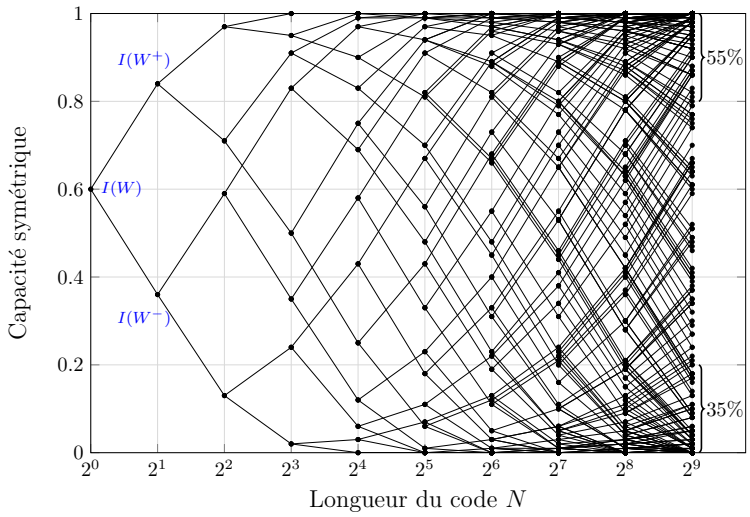


FIGURE 1.3: *Valeurs des capacités symétriques des canaux virtuels pour un canal BEC de probabilité d'effacement* $p = 0.4$ *(*$I(W) = 0.6$*)*

de cette distribution est donnée dans la figure 1.3 pour un canal BEC (d'après les équations (1.9) et (1.10)), et permet de mettre en évidence la concentration des valeurs autour de 0 et 1.

La propriété (1.7) démontrée au niveau du noyau se généralise à l'ensemble des canaux virtuels par:

$$\mathbb{E}(I(W_N^{(i)})) = I(W) \quad \forall N = 2^n, n \in \mathbb{N} \tag{1.12}$$

En exploitant cette propriété, Arikan a démontré dans [5] que le processus stochastique $(I_n)_{n\geq 0}$, avec $I_n = \{I(W_N^{(i)}), \forall i \in \{0,\dots,N-1\}, N = 2^n\}$ est en réalité une martingale. Dès lors, le théorème de convergence indique qu'il existe une variable aléatoire I_∞ telle que $(I_n)_{n\geq 0}$ converge vers I_∞ presque sûrement. Arikan a démontré de plus que I_∞ est donnée par:

$$I_\infty = \begin{cases} 1 & \text{avec probabilité } \ I(W) \\ 0 & \text{avec probabilité } 1 - I(W) \end{cases} \tag{1.13}$$

Le même résultat s'applique à la suite des $(Z_n)_{n\geq 0}$, où $Z_n = \{Z(W_N^{(i)}), \forall i \in \{0, N-1\}, N = 2^n\}$, à ceci près que, d'après (1.4), $\Pr(Z_\infty = 1) = 1 - I(W)$ tandis que $\Pr(Z_\infty = 0) = I(W)$. Ce résultat fondamental, connu sous le nom de *phénomène de polarisation*, indique que les canaux virtuels tendent asymptotiquement à devenir soit totalement bruités (capacité symétrique nulle) soit parfaits (capacité symétrique égale à 1). De plus, la fraction de canaux parfaits tend vers $I(W)$, ce qui prouve qu'il est possible de transmettre avec une probabilité d'erreur arbitrairement faible pour tout rendement inférieur ou égal à $I(W)$. C'est cette dernière propriété qu'exploitent les codes polaires afin d'atteindre la capacité de Shannon.

A noter que dans la formalisation initiale de la polarisation dans [5], la matrice de transformation est en réalité définie par $F_N = B_N \cdot G_N$, où B_N représente la permutation par inversion de bit (permutation "bit-reversal" en anglais). Par rapport à la description donnée ci-dessus, il ne s'agit que d'une permutation d'indices, donc sans impact sur les propriétés associées.

Il faut finalement mentionner que le phénomène de polarisation n'est pas spécifique au noyau G_2 proposé par Arikan, mais apparaît pour toute matrice carrée $A = (a_{i,j})_{0\leq i\leq \ell-1, 0\leq j\leq \ell-1}$, inversible et non triangulaire supérieure après toute permutation de colonnes, en appliquant récursivement le produit de Kronecker [55]. Ce chapitre se concentrera sur les codes polaires définis à partir du noyau G_2, appelé "noyau d'Arikan", tandis que l'utilisation d'autres noyaux [60, 17, 44], ou la combinaison de différents noyaux [76, 16] sera considérée dans le chapitre suivant.

1.2 Famille des codes polaires

1.2.1 Définition des codes polaires

Un code polaire est défini par le triplet $(N, K, \mathcal{I})$ où $N = 2^n$ correspond au nombre de bits codés, K au nombre de bits d'information et $\mathcal{I} = \{i_0, \dots, i_{K-1}\}$ est un ensemble d'indices $0 \leq i_k < N$ servant à indiquer la position des bits d'information parmi le vecteur de longueur N. La séquence des bits d'information est notée $u_{\mathcal{I}} = \{u_i, i \in \mathcal{I}\}$, tandis que la séquence $u_{\mathcal{I}^c} = \{u_i, i \in \mathcal{I}^c\}$, où $\mathcal{I}^c$ est le complémentaire de $\mathcal{I}$ dans $\{0, \dots, N-1\}$, est appelée la séquence des *bits figés*, dont les valeurs sont fixées et connues au niveau de l'encodeur et du décodeur. L'opération d'encodage d'un code polaire s'écrit $x_0^{N-1} = u_0^{N-1} \cdot G_N$, où $G_N = G_2^{\otimes n}$ est la matrice de polarisation telle que définie dans la partie précédente. Cette opération peut se décomposer comme suit:

$$\begin{aligned} x_0^{N-1} &= u_0^{N-1} \cdot G_N \\ &= u_{\mathcal{I}} \cdot G_N(\mathcal{I}) + u_{\mathcal{I}^c} G_N(\mathcal{I}^c), \end{aligned}$$

où $G_N(\mathcal{I})$ et $G_N(\mathcal{I}^c)$ correspondent aux sous-matrices de G_N obtenues en ne conservant que les lignes dans $\mathcal{I}$ et $\mathcal{I}^c$ respectivement. Il apparaît dès lors que la séquence $u_{\mathcal{I}^c}$ ne fait qu'ajouter à tous les mots de code la séquence $u_{\mathcal{I}^c} G_N(\mathcal{I}^c)$. Elle ne modifie donc pas les propriétés du code, pas plus qu'elle n'a d'impact sur les performances de décodage. On adoptera donc la convention la plus répandue, à savoir mettre à 0 tous les bits figés ($u_{\mathcal{I}^c} = \mathbf{0}_{N-K}$).

En toute rigueur, la définition ci-dessus ne caractérise pas la famille des codes polaires, mais une famille plus large où n'importe quel ensemble $\mathcal{I}$ est envisageable, et qui n'atteint pas la capacité de Shannon. Pour définir une famille atteignant asymptotiquement la capacité d'un canal W, il est nécessaire d'ajouter une condition sur $\mathcal{I}$, garantissant l'utilisation des canaux virtuels les plus fiables pour transmettre les bits d'information, et formulée par Arikan comme suit [5]:

$$\forall i \in \mathcal{I},\ \forall j \in \mathcal{I}^c, \qquad Z(W_N^{(i)}) \leq Z(W_N^{(j)}) \tag{1.14}$$

L'opération consistant à déterminer $\mathcal{I}$ suivant ce critère est appelée la "construction du code" et les principales méthodes pour cela seront passées en revue dans la partie suivante. Auparavant, il est important de remarquer que ce critère dépend du canal W, de sorte qu'un code polaire est optimisé pour un canal donné et il n'y a aucune garantie qu'il soit également optimal pour un autre canal. Cependant, il a été mis en évidence le fait que l'ensemble $\mathcal{I}$ respecte au moins deux conditions, indépendamment du canal W. Ces propriétés universelles reposent sur le fait qu'il existe un *ordre partiel* des canaux virtuels, à savoir qu'il existe des indices i et j tels que les canaux virtuels associés vérifient $Z(W_N^{(i)}) \leq Z(W_N^{(j)})$ indépendamment du canal W.

1.2.2 Ordre partiel des canaux virtuels

Pour formaliser la règle relative à l'ordre partiel des canaux virtuels, il convient au préalable d'expliciter le lien bien connu entre un canal virtuel d'indice $i \in \{0, \dots, N-1\}$ et le développement binaire de i, noté par $B_n^{(i)} = (b_n^{(i)}[0], \dots, b_n^{(i)}[n-1])$ avec $i = \sum_{k=0}^{n-1} b_n^{(i)}[k] \cdot 2^{n-1-k}$ ($b_n^{(i)}[0]$ est le bit de poids fort). Pour cela, à partir de la figure 1.2, il peut être observé que le bloc correspondant aux indices $i \in \{0, \dots, N/2-1\}$, dont le bit de poids fort $b_n^{(i)}[0]$ est égal à 0, est associé au canal W^-, tandis que celui correspondant aux indices $i \in \{N/2, \dots, N\}$, dont le bit de poids fort $b_n^{(i)}[0]$ est égal à 1, est associé à W^+. Plus généralement, on peut vérifier par récurrence que le développement binaire $B_n^{(i)}$ fait office de fonction indicatrice sur les opérations appliquées sur le canal virtuel $W_N^{(i)}$: dégradation si $b_n^{(i)}[k] = 0$ ou amélioration si $b_n^{(i)}[k] = 1$.

La conséquence immédiate est que, pour un canal W quelconque:

$$b_n^{(i)}[k] \geq b_n^{(j)}[k], \forall k \in \{0,\dots,n-1\} \Rightarrow Z(W_N^{(i)}) \leq Z(W_N^{(j)}) \tag{1.15}$$

Cette règle permet d'établir un ordre partiel entre deux indices dont le poids du développement binaire est distinct. Il existe une deuxième règle établissant un ordre partiel entre des indices dont le développement binaire est de même poids, appelée parfois "left-swap", et traduisant le fait que le canal obtenu à l'issue d'une amélioration puis d'une dégradation (en partant du canal W) est plus fiable que lorsque l'ordre des opérations est inversé [87, 43, 9]. Celle-ci peut se décrire par:

$$\left.\begin{array}{ll} \exists k_1 \in \{0,\dots,n-1\} \text{ tel que} & b_n^{(i)}[k_1] = 1 \text{ et } b_n^{(j)}[k_1] = 0 \\ \exists k_2 > k_1 \quad \text{tel que} & b_n^{(i)}[k_2] = 0 \text{ et } b_n^{(j)}[k_2] = 1 \\ \forall k \in \{0,\dots,n-1\}\backslash\{k_1,k_2\}, & b_n^{(i)}[k] = b_n^{(j)}[k] \end{array}\right\} \Rightarrow Z(W_N^{(i)}) \leq Z(W_N^{(j)}) \tag{1.16}$$

Précisons que ce cas de figure peut se présenter plusieurs fois dans le développement binaire de i et j sans changer la conclusion (pour autant que le "1" se présente toujours d'abord dans le développement de i). Il faut également prendre en compte la combinaison de ces deux règles. Nous proposons la formalisation suivante pour caractériser l'ordre partiel, incluant toute combinaison des deux règles précédentes:

Proposition 1 *Soit un canal W. Pour tout $(i,j) \in \{0,\dots,N-1\}^2$*

$$\sum_{k=0}^{m} b_n^{(i)}[k] \geq \sum_{k=0}^{m} b_n^{(j)}[k], \quad \forall m \in \{0,\dots,n-1\} \Rightarrow Z(W_N^{(i)}) \leq Z(W_N^{(j)}) \tag{1.17}$$

La justification de cette proposition est donnée en Annexe A.1. Dans la suite, on notera par $i \succeq j$ le fait que deux indices i et j sont liés par l'équation (1.17). Dans ce cas, on a nécessairement :

$$j \in \mathcal{I} \Rightarrow i \in \mathcal{I}$$
$$i \notin \mathcal{I} \Rightarrow j \notin \mathcal{I}$$

On notera qu'on ne peut avoir $i \succeq j$ que si $|B_n^{(i)}| \geq |B_n^{(j)}|$, dans la mesure où $|B_n^{(i)}| = \sum_{k=0}^{n-1} b_n^{(i)}[k]$ et $|B_n^{(j)}| = \sum_{k=0}^{n-1} b_n^{(j)}[k]$. Dans le cas où la condition (1.17) n'est pas vérifiée entre deux canaux virtuels, cela signifie que la hiérarchie entre les deux dépend du canal W considéré et qu'il n'est pas possible de les ordonner a priori. A titre d'exemple, le tableau 1.1 fournit le cas d'un code $N = 8$. Il peut être observé que la propriété (1.17) est toujours vérifiée lorsque $i > j$ sauf lorsque $i = 4$ et $j = 3$. Dès lors, il n'existe que deux ordres possibles pour les capacités symétriques des canaux virtuels : $\{0,1,2,3,4,5,6,7\}$ ou $\{0,1,2,4,3,5,6,7\}$. La discrimination entre les deux étant dépendante du canal W.

La connaissance de l'ordre partiel peut servir à simplifier la recherche de l'ensemble $\mathcal{I}$ pour un canal donné. Il a été montré qu'il peut suffire de calculer de l'ordre de $N/\log_{3/2}(N)$ coefficients $Z(W_N^{(i)})$ pour identifier les meilleurs canaux virtuels [64]. Il est aussi possible de représenter l'ensemble des relations d'ordre entre les canaux virtuels sur un diagramme de Hasse [87], et d'expliciter la construction de ce diagramme de manière récursive [43].

Au delà de la simplification de la construction de l'ensemble $\mathcal{I}$, cet ordre partiel présente également un intérêt pour le décodage. En mettant en lumière le fait que certains canaux virtuels sont nécessairement moins fiables que d'autres, l'ordre partiel permet d'identifier des positions critiques, nettement plus susceptibles de fausser le décodage [111].

Finalement, et c'est certainement la conséquence la plus importante, l'ordre partiel confère à la famille des codes polaires des propriétés algébriques singulières, permettant d'expliciter analytiquement un certain nombre de propriétés relatives au spectre des distances de ces codes.

TABLE 1.1: *Détermination de l'ordre partiel des canaux virtuels pour* $N = 8$

Indice	Développement binaire	Somme cumulée
0	000	000
1	001	001
2	010	011
3	011	**012**
4	100	**111**
5	101	112
6	110	122
7	111	123

1.2.3 SPECTRE DES DISTANCES DES CODES POLAIRES

La distance minimale d'un code obtenu à partir de la matrice G_N (sans contrainte particulière sur $\mathcal{I}$) est donnée par [46]:

$$D_{\min} = \min_{i \in \mathcal{I}} w_h(i), \tag{1.18}$$

où $w_h(i)$ est le poids de hamming de la i-ième ligne de G_N. Il peut de plus être montré que $w_h(i) = 2^{|B_n^{(i)}|}$ où $|B_n^{(i)}| = \sum_{k=0}^{n-1} b_n^{(i)}[k]$ est le poids de hamming de $B_n^{(i)}$. En revanche, il n'existe pas de formule analytique pour déterminer le nombre de mots de code de poids faible pour un cas aussi général. Cependant, dans [9], les auteurs se sont intéressés à la famille des codes issus de la matrice G_N et vérifiant la condition d'ordre partiel, appelée famille des codes monomiaux décroissants (CMD), et incluant la famille des codes polaires.

Définition 1 *Un code $\mathcal{C}(N,K,\mathcal{I})$ est dit CMD si les deux conditions suivantes sont satisfaites:*

- *si $j \in \mathcal{I}$ et $i \succeq j \Rightarrow i \in \mathcal{I}$*
- *si $i \notin \mathcal{I}$ et $i \succeq j \Rightarrow j \notin \mathcal{I}$*

Il est révélé dans [9] que les CMD disposent d'une structure algébrique singulière, permettant d'expliciter analytiquement le nombre de mots de code de poids faible, à partir de la dimension des diagrammes de Ferrer des monomiaux. En combinant des résultats de [9], nous avons simplifié la formule proposée, de manière à ne faire intervenir que le développement binaire des indices $i \in \mathcal{I}$:

Proposition 2 *Le nombre de mots de code de poids faible d'un CMD $\mathcal{C}(N,K,\mathcal{I})$ de distance minimale $D_{min} = 2^d$ est donné par:*

$$\mu_{min} = 2^{n - \frac{d(d+1)}{2}} \sum_{\substack{i \in \mathcal{I} \\ |B_n^{(i)}| = d}} 2^{s_n(i)}, \tag{1.19}$$

avec

$$s_n(i) \triangleq \sum_{j=0}^{n-1} j \cdot b_n^{(i)}[j] \tag{1.20}$$

La démonstration est fournie en Annexe A.2.1. Par ailleurs, le nombre de mots de code de poids faible d'un CMD $\mathcal{C}(N,K,\mathcal{I})$ peut également se calculer de manière récursive, à partir de ceux des *codes composites*. Par

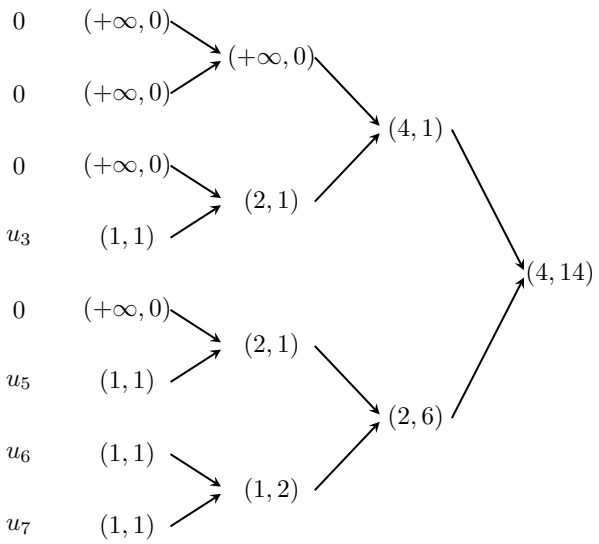


FIGURE 1.4: *Calcul du nombre de mots de code de poids faibles par accumulation*

"codes composites", on entend les deux codes $\mathcal{C}_1(N/2,K_1,\mathcal{I}_1)$ et $\mathcal{C}_2(N/2,K_2,\mathcal{I}_2)$ tels que $K = K_1 + K_2$ et:

$$\mathcal{I} = \mathcal{I}_1 \cup \{i + N/2, i \in \mathcal{I}_2\} \tag{1.21}$$

Il peut être montré que les codes composites $\mathcal{C}_1$ et $\mathcal{C}_2$ d'un CMD sont eux-mêmes des CMD, et sont tels que $D_1 \geq D_2$, où D_1 et D_2 sont les distances minimales de $\mathcal{C}_1$ et $\mathcal{C}_2$ respectivement (cf Annexe A.2.2).

Proposition 3 *Soit un CMD $\mathcal{C}(N,K,\mathcal{I})$, dont les codes composites sont $\mathcal{C}_1(N,K_1,\mathcal{I}_1)$ et $\mathcal{C}_2(N,K_2,\mathcal{I}_2)$ de distances minimales respectives D_1 et D_2 et de multiplicités μ_1 et μ_2, alors le code $\mathcal{C}$ est tel que:*

$$\begin{cases} D_{min} = D_1 \ \ et \ \ \mu_{min} = 2 \cdot D_1 \cdot \mu_1 & si \ \ D_1 = D_2 \\ D_{min} = 2D_2 \ \ et \ \ \mu_{min} = \mu_2 + 2 \cdot D_1 \cdot \mu_1 & si \ \ D_1 = 2D_2 \\ D_{min} = 2D_2 \ \ et \ \ \mu_{min} = \mu_2 & si \ \ D_1 > 2D_2 \end{cases} \tag{1.22}$$

A partir de cette propriété, le nombre de mots de code de poids faible peut se calculer par accumulation au fur et à mesure des étapes de récursion. L'initialisation peut se faire à partir de $N = 1$ d'après:

$$(D_i,\mu_i) = \begin{cases} (+\infty,0) & \text{si } \ i \notin \mathcal{I} \\ (1,1) & \text{si } \ i \in \mathcal{I} \end{cases} \tag{1.23}$$

Un exemple du processus est proposé dans la figure 1.4 pour un code $\mathcal{C}(8,4,\{3,5,6,7\})$, pour lequel on obtient $D_{\min} = 4$ et $\mu_{min} = 14$.

Nous proposons également un corollaire de la proposition 2, caractérisant, pour tout couple (N,K), le code maximisant la distance minimale et minimisant la multiplicité:

Proposition 4 *Pour tout $(N = 2^n,K)$, le CMD maximisant la distance minimale et minimisant la multiplicité est caractérisé par:*

$$\forall i \in \mathcal{I}, \forall j \notin \mathcal{I} \Rightarrow \begin{cases} |B_n^{(i)}| > |B_n^{(j)}| \\ ou \ \ |B_n^{(i)}| = |B_n^{(j)}| \ et \ \ s_n(i) \leq s_n(j) \end{cases} \tag{1.24}$$

La première condition garantit de maximiser la distance minimale et la deuxième de minimiser la multiplicité. Une condition importante à vérifier est que le code obtenu est bien CMD pour pouvoir appliquer la proposition 2. Cette démonstration est donnée en Annexe A.2.3. Dans la suite, nous appellerons code SDO (pour Spectrum Distance Optimal) tout code vérifiant la proposition 4. Un code SDO s'obtient en sélectionnant les lignes par poids décroissant, puis pour un même poids par $s_n(i)$ croissant. Il convient de préciser que, pour un couple (N,K) donné, il peut exister plusieurs codes SDO, ayant la même distance minimale et le même nombre de mots de code de poids minimal [2]. Si l'on ajoute comme contrainte de trouver le meilleur code SDO pour le décodeur par annulation successive (cf section suivante) sur un canal W donné, il est possible d'utiliser les méthodes de construction d'un code polaire, détaillées dans la section 1.3.2, pour discriminer dans les cas où $|B_n^{(i)}| = |B_n^{(j)}|$ et $s_n(i) = s_n(j)$.

Il convient de mentionner un cas très particulier de CMD (et également SDO), à savoir les codes Reed-Muller (RM), inventés en 1954 [71]. Un code RM est défini par deux paramètres r et n appelés ordre et nombre de variables respectivement, dont la longueur est donnée par $N = 2^n$, le nombre de bits d'information par $K = \sum_{k=0}^{r} \binom{n}{k}$ et la distance minimale par 2^{n-r}. Il a été montré qu'il peuvent s'obtenir à partir de la matrice G_N en sélectionnant $\mathcal{I}$ d'après le critère suivant [35]:

$$\mathcal{I}_{RM}^{(n,r)} = \{i \in \{0,\dots,N-1\} \mid |B_n^{(i)}| \geq n-r\} \tag{1.25}$$

Les codes RM font partie des familles de codes dont les propriétés algébriques sont les mieux connues, et le rapprochement avec les codes CMD est tout à fait intéressant. Considérons un CMD $\mathcal{C}(N = 2^n, K, \mathcal{I})$ et le code $\mathrm{RM}(n,r)$ de même distance minimale, tel que $\mathcal{I} \subseteq \mathcal{I}_{RM}^{(n)}$. Dès lors, il est clair que :

$$X \in \mathcal{C} \Rightarrow X \in \mathrm{RM}(n,r). \tag{1.26}$$

Or, il est connu de longue date que pour un code RM, les mots de code de poids compris entre $D_{\min}$ et $2D_{\min}$ ont tous des poids de la forme $2D_{\min} - 2^i$ [13]. Cette propriété se transmet directement aux CMD, et permet de prouver, en particulier, qu'il n'existe pas de mots de code de poids compris strictement entre $D_{\min}$ et $\frac{3}{2}D_{\min}$ (sachant que $D_{\min}$ est une puissance de 2).

Un résultat significatif sur le lien entre les codes polaires et RM est donné dans [65], prouvant que lorsque le canal W tend vers un canal parfait, et sous réserve que le rendement coïncide avec celui d'un code RM, alors le code polaire tend vers un code RM. Une extrapolation directe de cette propriété confirme que la distance minimale d'un code polaire tend vers la valeur maximale possible pour le couple (N,K) quand le canal tend vers le canal parfait.

Finalement, la connaissance du nombre de mots de code de poids faible permet d'obtenir une estimation de la borne de l'union (UB pour union bound, en anglais), borne supérieure de la performance maximum de vraisemblance (ML pour Maximum Likelihood, en anglais). En particulier, sur un canal BI-AWGN (pour Binary-Input Additive white Gaussian noise, en anglais) de variance σ, celle-ci se calcule par :

$$U_b(\sigma) = \sum_{d=D_{\min}}^{N} \mu_d Q\left(\frac{\sqrt{d}}{\sigma}\right), \tag{1.27}$$

où μ_d correspond au nombre de mots de code de poids d, et Q est reliée à la fonction d'erreur gaussienne complémentaire par $Q(x) = \frac{1}{2}\mathrm{erfc}(\frac{x}{\sqrt{2}})$. Une approximation classique est de ne conserver que le terme dominant:

$$U_b(\sigma) \approx \overline{U_b}(\sigma) = \mu_{D_{\min}} Q\left(\frac{\sqrt{D_{\min}}}{\sigma}\right). \tag{1.28}$$

[2] Il n'y a en revanche aucune garantie d'égalité concernant le nombre de mot de code de poids $D > \mathrm{D}_{\min}$.

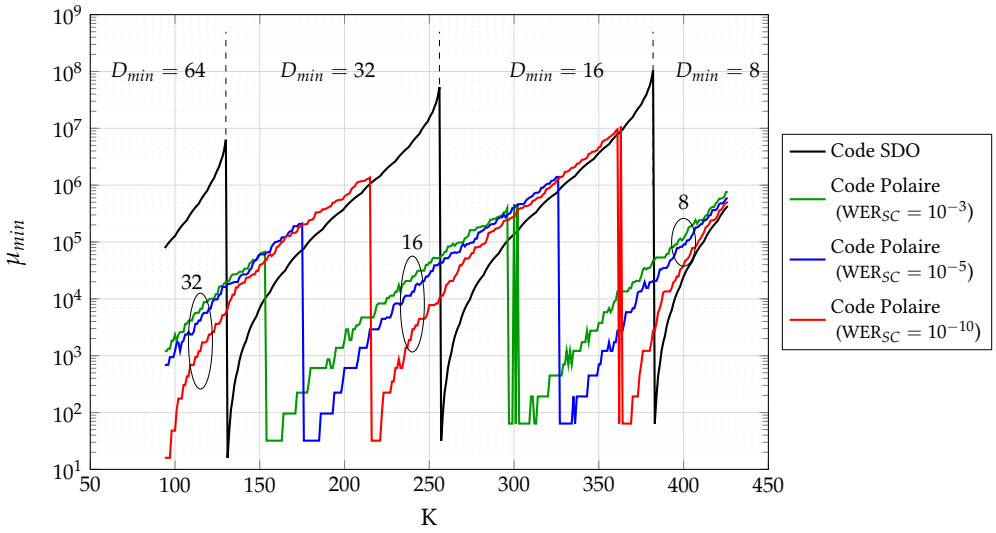


FIGURE 1.5: *Multiplicité minimale des codes polaires* $N = 512$

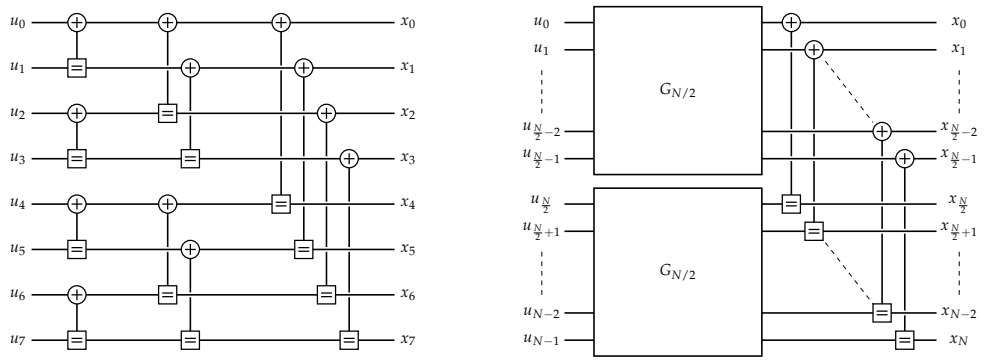
Cette approximation est d'autant plus précise pour les CMD, qu'il a été montré que le prochain terme (éventuellement) non nul correspond à $d = \frac{3}{2}D_{\min}$. Par la suite, lorsqu'il sera question de tracer la borne de l'union, celle-ci sera systématiquement estimée à partir de cette dernière approximation.

En guise de conclusion de cette section, la figure 1.5 présente, pour $N = 512$ et tout rendement $1/6 \leq R \leq 5/6$, la multiplicité minimale du code SDO ainsi que des codes polaires construits en utilisant l'approximation gaussienne (cf Section 1.3.2) aux valeurs de SNR telles que le taux d'erreur paquet du décodage SC, noté WER_{SC}, soit égal à 10^{-3}, 10^{-5} et 10^{-10} respectivement. Les ruptures des différentes courbes correspondent aux changements de distance minimale des codes. Il apparaît que le code polaire se rapproche bien du code SDO lorsque le SNR de construction augmente. Cependant, ce rapprochement est assez lent et ne survient que pour des taux d'erreurs excessivement faibles, de sorte que les codes polaires utilisés en pratique ont une distance minimale relativement basse comparée à celle du code SDO. On notera toutefois que pour certaines valeurs de rendements, la différence entre les deux codes, tant en termes de distance minimale que de multiplicité, peut être assez réduite. Dans ces cas là, on verra que le code SDO peut être utilisé pour améliorer les performances avec certains décodeurs polaires (cf section 1.4).

1.3 Décodage par annulation successive et construction du code

1.3.1 Décodage par annulation successive

Si l'opération d'encodage d'un code polaire peut aussi bien s'exprimer par le produit matriciel avec G_N que par la propagation des bits d'informations sur le graphe du code (tel que celui présenté sur la figure 1.6), le processus de décodage s'exprime pour sa part exclusivement sur ce dernier. La différence étant que l'encodage est une propagation de la gauche vers la droite de la séquence u_0^{N-1}, tandis que le décodage part des valeurs reçues y_0^{N-1} à droite du graphe et propage des rapports de vraisemblance logarithmiques (LLR pour Log-likelihood Ratio, en anglais) de manière à déterminer les estimations des bits d'information $\hat{u}_{\mathcal{I}}$ à gauche du graphe. Ce graphe de décodage contient deux types de nœuds: les *nœuds xor*, de symboles $\oplus$, correspondant à l'opérateur logique "ou exclusif", et les *nœuds répétitions*, de symboles $\boxed{=}$, décrivant l'égalité entre toutes les entrées et sorties. Un exemple de graphe est fourni sur la figure 1.6, présentant le



(a) Graphe $N = 8$ (b) Construction récursive

FIGURE 1.6: *Graphe d'un code polaire de dimension $N = 8$ (a) et transformation récursive (b)*

cas $N = 8$, ainsi que la transformation récursive permettant de générer le graphe pour toute valeur de N.

L'initialisation du décodeur consiste à calculer les LLRs d'entrée à partir des valeurs reçues:

$$L_{in}^{(i)} = \log\left(\frac{\Pr(x_i = 0 \,|\, y_i)}{\Pr(x_i = 1 \,|\, y_i)}\right) \tag{1.29}$$

Dans le décodage par *annulation successive* (SC, pour successive cancellation, en anglais), fondé sur le phénomène de polarisation, les bits d'information sont décodés les uns après les autres, par ordre croissant d'indice, et chaque estimation est utilisée pour décoder les bits suivants[3]. Étant donné un bit u_i, et connaissant les estimations des bits d'information précédemment décodés $\hat{u}_0^{i-1}$, le décodeur procède au calcul du LLR du bit:

$$L_i(y_0^{N-1}, \hat{u}_0^{i-1}) = \log\left(\frac{\Pr(u_i = 0 \,|\, y_0^{N-1}, \hat{u}_0^{i-1})}{\Pr(u_i = 1 \,|\, y_0^{N-1}, \hat{u}_0^{i-1})}\right), \tag{1.30}$$

puis prend la décision dure $\hat{u}_i$ pour le bit en question d'après:

$$\hat{u}_i = h_{\mathcal{I}}(L_i) \overset{\text{def}}{=} \begin{cases} u_i & \text{if } i \notin \mathcal{I} \\ \frac{1-\text{sign}(L_i)}{2} & \text{if } i \in \mathcal{I}, \end{cases} \tag{1.31}$$

où $L_i \triangleq L_i(y_0^{N-1}, \hat{u}_0^{i-1})$. Ce processus est répété successivement pour chaque bit d'information, comme présenté dans l'algorithme 1.

La propagation des LLRs dans le graphe suit deux règles distinctes, relatives à chacun des deux types de nœuds. La figure 1.7 décrit le sens de propagation des messages, depuis les LLRs d'entrée L_a et L_b vers les sorties $L_{out}^{(xor)}$ et $L_{out}^{(rep)}$ respectivement. Les équations de mise à jour sont:

$$L_{out}^{(xor)} \overset{\text{BP}}{=} 2 \cdot \text{atanh}\left(\tanh\left(\frac{L_a}{2}\right) \cdot \tanh\left(\frac{L_b}{2}\right)\right) \tag{1.32}$$

$$L_{out}^{(xor)} \overset{\text{MS}}{\approx} \text{sign}(L_a \cdot L_b) \cdot \min(L_a, L_b) \tag{1.33}$$

$$L_{out}^{(rep)} = (-1)^v L_a + L_b \tag{1.34}$$

avec v correspondant à la valeur du bit de l'arrête en question. Cette valeur est issue de la repropagation dans le graphe des décisions précédentes $\hat{u}_0^{i-1}$ où i est l'indice du bit dont on cherche l'estimation.

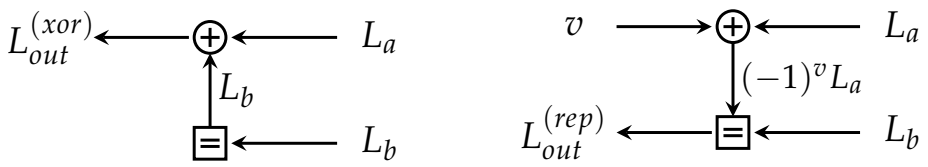
[3]De la même manière que dans la polarisation étudiée dans la section précédente, les canaux virtuels étaient définis connaissant les valeurs des bits précédents u_0^{i-1}.

Algorithme 1 Décodage par annulation successive d'un code polaire $\mathcal{C}(N,K,\mathcal{I})$

procédure SC(y_0^{N-1})
  Initialisation : calculer $L_{in}^{(i)}, \forall i \in \{0,\dots,N-1\}$
  pour $i = 0,\dots,N-1$ **faire**
    Calculer $L_i(y_0^{N-1}, \hat{u}_0^{i-1})$;
    Déterminer $\hat{u}_i = h_{\mathcal{I}}(L_i)$;
  fin pour
retourne $\hat{u}_{\mathcal{I}}$;
fin procédure

Algorithme 2 Calcul du LLR du bit u_i

procédure CALCLLR(i , $(\lambda_0^{(0)}, \lambda_1^{(0)}, \dots, \lambda_{N-1}^{(0)}), \hat{u}_0^{i-1}$)
  pour $\ell = 1,\dots,n$ **faire**
    si $b_n^{(i)}[\ell-1] = 0$ **alors** ▷ Nœuds xor
      pour $k = 0,\dots,2^{n-\ell}-1$ **faire**
        $\lambda_k^{(\ell)} = \mathrm{BP}(\lambda_k^{(\ell-1)}, \lambda_{k+2^{n-1-\ell}}^{(\ell-1)})$;
      fin pour
    sinon ▷ Nœuds répétition
      $v = (\hat{u}_{(\lfloor i\cdot 2^{\ell-n}\rfloor-1)\cdot 2^{n-\ell}}, \dots, \hat{u}_{(\lfloor i\cdot 2^{\ell-n}\rfloor)\cdot 2^{n-\ell}-1}) \cdot G_2^{\otimes n-\ell}$ ▷ Repropagation des décisions
      pour $k = 0,\dots,2^{n-\ell}-1$ **faire**
        $\lambda_k^{(\ell)} = (-1)^{v(k)} \cdot \lambda_k^{(\ell-1)} + \lambda_{k+2^{n-1-\ell}}^{(\ell-1)}$;
      fin pour
    fin si
  fin pour
retourne $\lambda_0^{(n)}$;
fin procédure



(a) Nœud xor (b) Nœud répétition

FIGURE 1.7: *Règles de propagation des LLRs pour les nœuds xor (a) et répétition (b)*

A noter que la fonction de mise à jour du nœud xor existe sous forme exacte appelée BP (pour Belief-propagation, en anglais, terminologie empruntée aux codes LDPC) faisant intervenir les fonctions complexes tangente hyperbolique et sa réciproque, ou sous forme approchée MS (pour Min-Sum), nettement plus adaptée pour les implémentations matérielles. Si cette approximation peut générer une dégradation des performance par rapport à la formule exacte du BP pour les codes LDPC, il a été observé que cette différence n'est pas significative dans le cas des codes polaires [59].

Le détail du processus utilisé pour calculer le LLR du bit u_i est donné dans l'algorithme 2, où $\lambda_k^{(0)}$ représente le k-ième LLR d'entrée (*i.e.* $L_{in}^{(k)}$), et $\lambda_k^{(\ell)}$ la valeur d'un LLR après la propagation à travers

ℓ noyaux; le LLR final $L_i(y_0^{N-1}, \hat{u}_0^{i-1})$ est donné par $\lambda_0^{(n)}$. Le type de nœud traversé est indiqué par le développement binaire de i, à savoir xor si $b_n^{(i)}[\ell] = 0$ et répétition sinon.

La complexité d'un algorithme de décodage s'évalue sommairement à partir du nombre de calculs de LLRs. Dans le cas du décodage SC, cette complexité est nettement réduite grâce à la ré-utilisation des calculs intermédiaires. Pour cela, il convient de noter qu'à chaque arête horizontale du graphe est associé un et un seul LLR durant tout le processus de décodage. Dès lors, le nombre total de LLRs à calculer est donné par le nombre d'arêtes horizontales, à savoir $N \cdot n$. La complexité du décodage SC est donc de l'ordre de $\mathcal{O}(N \log(N))$. Cette réduction de complexité peut s'observer dans l'algorithme 2 en considérant le calcul des LLRs de deux bits distincts u_i et u_j: tant que le développement binaire des indices i et j est identique, les calculs intermédiaires restent exactement les mêmes dans les deux cas, de sorte qu'il suffit de stocker les résultats en mémoire lors du décodage de u_i et de les réutiliser lors du décodage du bit u_j.

Une propriété remarquable du décodeur SC est sa flexibilité en fonction du rendement. En effet, l'ensemble $\mathcal{I}$ peut être vu comme un paramètre dans l'algorithme 1 de telle sorte qu'un seul décodeur est capable de fonctionner pour tout $K \in \{0, \ldots, N-1\}$. Cette aptitude est hautement intéressante dans les applications pratiques tolérant différents schémas de modulation codée (MCS, de l'anglais Modulation Coding Scheme) ayant chacun un rendement différent. Il convient cependant de souligner qu'il est délicat d'obtenir une implémentation supportant toute valeur de rendement, et étant en même temps excellente en termes de latence. Ainsi, les implémentations des codes polaires sacrifient le plus souvent la flexibilité au profit de la latence (cf section 1.4.4).

1.3.2 Construction d'un code polaire

On appelle « construction » d'un code polaire l'opération ayant pour objectif, connaissant le canal W, de déterminer l'ensemble $\mathcal{I}$, *i.e.* les K canaux virtuels les plus fiables. Il existe dans la littérature deux principales approches pour traiter ce problème. La première, proposée initialement par E. Arikan [5], repose sur une optimisation heuristique basée sur des simulations Monte-Carlo. La deuxième est basée sur le concept d'évolution de densité, très utilisé notamment pour les codes LDPC [80]. Il existe cependant une troisième approche [43], dont le but est de fournir une solution mathématique simple et immuable, mais sans prise en compte rigoureuse du canal.

1.3.2.1 Méthode heuristique

Le grand avantage de cette méthode est la simplicité de sa mise en œuvre, puisqu'il suffit d'implémenter l'encodeur et le décodeur pour pouvoir l'appliquer à tout modèle de canal. Il s'agit d'accumuler, en générant un grand nombre de réalisations de bruit, des données statistiques relatives aux différents canaux virtuels de manière à sélectionner, in fine, les plus fiables. Pour cela, il convient d'utiliser un décodeur *genie-aided* (traduit littéralement "aidé par un génie") calculant pour chaque indice $i \in \{0, \ldots, N-1\}$ le LLR *genie-aided* en connaissant les vraies valeurs des bits précédents:

$$L_i^{(ga)}(y_0^{N-1}, u_0^{i-1}) = \log\left(\frac{\Pr(u_i = 0 \,|\, y_0^{N-1}, u_0^{i-1})}{\Pr(u_i = 1 \,|\, y_0^{N-1}, u_0^{i-1})}\right). \tag{1.35}$$

Notons que, pou une réalisation de bruit donnée, L_i est rigoureusement égal à $L_i^{(ga)}$ dans le cas où tous les bits précédents sont ou figés ou correctement décodés. A partir de ces LLRs, il existe deux grandeurs statistiques utilisées pour évaluer la fiabilité des canaux virtuels :

- La probabilité d'erreur genie-aided: pour chaque canal virtuel, il s'agit de calculer la décision genie-aided $u_i^{(ga)} = (1 - \text{sign}(L_i^{(ga)}))/2$ et de la comparer avec la vraie valeur du bit u_i:

$$p_{i|} = \Pr(u_i^{(ga)} \neq u_i \,|\, y_0^{N-1}, u_0^{i-1}) \tag{1.36}$$

  Le choix des positions d'information est alors donné par les indices des K canaux virtuels ayant les plus faibles valeurs de $p_{i|}$. De plus, ces probabilités permettent de déterminer une borne supérieure de la probabilité d'erreur bloc du décodeur SC, notée WER_{SC} (pour Word Error Rate, en anglais), extrêmement précise pour les SNR modérés et élevés [70] :

$$\text{WER}_{SC} \leq \sum_{i \in \mathcal{I}} p_{i|} \tag{1.37}$$

- Le paramètre de Bhattacharyya : les valeurs des $Z(W_N^{(i)})$ peuvent s'obtenir d'après la relation [10]:

$$Z(W_N^{(i)}) = \mathbb{E}\left(\exp\left(\frac{1}{2}(2u_i - 1)L_i^{(ga)}\right)\right) \tag{1.38}$$

  Ce coefficient peut s'estimer par simulation Monte-Carlo en moyennant sur un grand nombre de réalisations de bruit. Par rapport au calcul des probabilités ci-dessus, il a l'avantage d'exploiter directement les informations souples, et a donc une meilleure stabilité numérique et une moins grande sensibilité au nombre total d'échantillons accumulés. En revanche, il ne permet pas d'obtenir une estimation précise des performances, mais seulement un encadrement [5, 55] :

$$\max_{i \in \mathcal{I}} \frac{1}{2}\left(1 - \sqrt{1 - Z(W_N^{(i)})^2}\right) \leq \text{WER}_{SC} \leq \sum_{i \in \mathcal{I}} Z(W_N^{(i)}) \tag{1.39}$$

1.3.2.2 Méthode par évolution de densité

La méthode par évolution de densité (DE pour Density Evolution en anglais) consiste à propager dans le graphe du code polaire non pas une séquence de LLRs (correspondant à une certaine réalisation du canal), mais directement la distribution de probabilité (PDF, pour Probability Density Function, en anglais) des LLRs, en utilisant un décodeur SC genie-aided. On suppose de plus que $x_0^{N-1} = u_0^{N-1} = \mathbf{0}_N$, ce qui n'a pas d'impact du fait de la symétrie du décodeur et du canal W. Il peut-être observé que, pour tout noyau du graphe, les entrées sont indépendantes de sorte qu'aucun biais n'est introduit en propageant la PDF. Une fois la distribution des LLRs obtenue pour chaque canal virtuel, la probabilité d'erreur $p_{i|}$ de l'équation (1.36) peut être calculée comme l'intégrale de la PDF sur $]-\infty, 0]$ (puisqu'un LLR négatif correspond à une mauvaise décision $\hat{u}_i$). Le point délicat de cette méthode est de déterminer les équations de propagation des PDFs. Si le nœud répétition se résume à une simple convolution des PDFs d'entrée, le calcul exact pour les nœuds xor est particulièrement complexe [80]. Il existe toutefois des approximations efficaces, notamment exploitant l'approximation Min-Sum [53]. A noter également qu'une convolution double le support des PDFs (discrétisées), de sorte qu'une approximation est nécessaire en pratique pour éviter l'explosion de la dimension du support des PDFs [94]. Dans le cas général d'un canal W, l'évolution de densité repose essentiellement sur une discrétisation des PDFs sur un très grand nombre de valeurs et reste exigeante en ressources de calcul, d'autant plus qu'elle doit être effectuée pour chaque canal W (y compris pour différentes valeurs de SNR). Il existe néanmoins des cas dans lesquels il est possible de s'en sortir à moindre frais.

Nous avons déjà vu que pour le canal BEC, il est possible de propager directement les valeurs des capacités symétriques dans le graphe d'après les équations (1.9) et (1.10). Si de telles équations n'existent

pas pour des modèles de canaux plus complexes, il existe une approximation pour le canal BI-AWGN permettant de simplifier considérablement le processus de construction, appelée *approximation gaussienne* (DE-GA pour Density Evolution using Gaussian Approximation en anglais). Cette fois-ci, les PDFs des canaux virtuels sont approximées par des distributions gaussiennes symétriques $\mathcal{N}(\mu,\sigma^2)$ avec $2\mu=\sigma^2$, et les PDFs sont alors décrites à partir d'un unique paramètre, généralement la moyenne μ [98, 110, 27]. Bien que fondée sur une hypothèse mathématiquement inexacte, la méthode DE-GA s'avère être un choix répandu en pratique, du fait que la différence avec l'évolution de densité exacte ou la méthode heuristique n'apparaît que pour des codes très longs ($n \geq 18$) [27].

A noter que si l'on suppose que toutes les distributions d'entrée du décodeur sont identiques, alors le nombre de PDFs distinctes à évaluer – indépendamment de la méthode utilisée – est $2N-2=\sum_{\ell=1}^{n} 2^\ell$, soit une complexité linéaire en la longueur du code [70]. Cela s'explique par le fait que, partant d'une unique PDF en entrée, il n'y a que 2 PDFs distinctes à l'issue du premier étage de noyaux, correspondant aux sorties des nœuds xor et répétition respectivement. Par récurrence, le nombre de PDFs distinctes après $\ell \in \{1,\dots,n\}$ étages de noyaux vaut exactement 2^ℓ.

Pour les simulations présentées dans la suite de ce manuscrit sur le canal BI-AWGN, chaque point de simulation utilisera le code construit pour la valeur courante de SNR par la méthode DE-GA.

1.3.2.3 Méthode arithmétique

La dernière approche, qualifiée ici d'arithmétique, vise à donner une caractérisation mathématique simple de l'ensemble $\mathcal{I}$. Elle a l'avantage d'être de faible complexité, facile d'implémentation et de fournir toujours rigoureusement la même solution. En contrepartie, le canal n'est pris en compte qu'à posteriori en vérifiant que la solution est efficace pour le décodage SC. Les positions d'information correspondent aux indices $i \in \{0,\dots,N-1\}$ maximisant la métrique suivante:

$$\Gamma(i) = \sum_{k=0}^{n-1} \left(\frac{1}{\beta}\right)^k \cdot b_n^{(i)}[k], \tag{1.40}$$

où β est une valeur à choisir judicieusement [4]. Il a été démontré que cette métrique respecte l'ordre partiel pour tout $\beta > 1$, et que, pour le canal BI-AWGN, $\beta = 1.1892$ offre des performances très comparables avec les méthodes par approximation gaussienne [43]. Nous mentionnons également qu'il est possible de relier ce critère avec celui permettant d'obtenir le code SDO:

Proposition 5 *Pour tout couple (N,K), le code SDO peut être obtenu en sélectionnant les K indices maximisant:*

$$\Gamma'(i) = \sum_{k=0}^{n-1} \left(1-\frac{k}{\nu}\right) \cdot b_n^{(i)}[k], \tag{1.41}$$

pour tout $\frac{n(n-1)}{2} \leq \nu < +\infty$.

La démonstration est donnée en Annexe A.2.4. On peut remarquer qu'en prenant $\beta^{-1} = 1-\frac{1}{\nu}$ et en appliquant un développement limité à l'ordre 1, la métrique (1.40) devient celle du code SDO (1.41), de sorte que le code SDO peut être vu comme un cas particulier de la méthode basée sur le développement β lorsque $\beta \to 1$.

La figure 1.8 présente l'impact de la valeur de β^{-1} sur les performances du code avec le décodeur SC d'une part et sur la borne de l'union d'autre part (calculée par l'équation (1.28)). Si l'on retrouve le fait que $\beta^{-1} = 1/1.1892 = 0.841$ donne effectivement la meilleure performance pour le décodage SC, il

[4] Dans [43], le développement binaire est défini avec $b_n^{(i)}[0]$ comme LSB, ce qui revient à remplacer β par $1/\beta$.

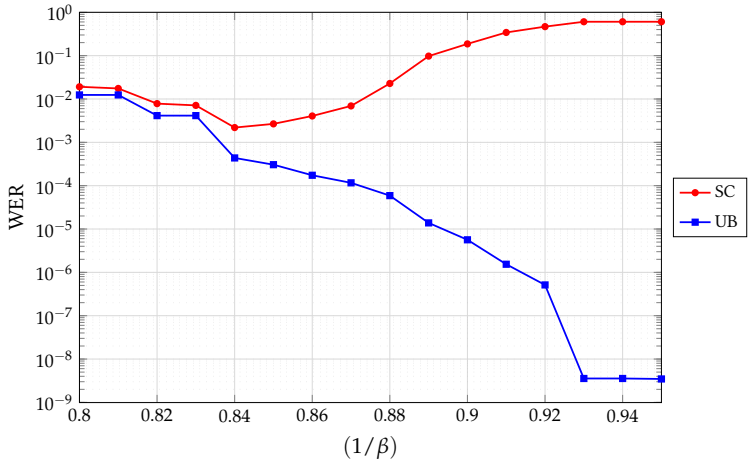


FIGURE 1.8: *Impact de la valeur β sur les propriétés du code $(N,K) = (512,256)$ à SNR$= 3dB$*

apparaît également que la borne de l'union continue de décroitre jusqu'à atteindre le seuil minimal correspondant au code SDO. Pour les paramètres présentés sur la figure, il est clair qu'il faudrait un décodeur excessivement puissant – et d'une complexité certainement impraticable – pour atteindre la borne ML du code SDO, sachant que le taux d'erreur du SC est supérieur à 10^{-1}. Néanmoins, on verra dans la suite de ce chapitre que pour certains rendements et avec le décodage par liste, le code SDO peut être largement avantageux. Pour les autres valeurs de rendement, il est nécessaire de trouver un compromis entre ces deux extrêmes, qui peut être obtenu en choisissant judicieusement la valeur de β, ou en utilisant un code polaire obtenu pour une valeur de SNR supérieure au SNR de fonctionnement.

1.4 Décodage des codes polaires

Si les codes polaires décodés avec le SC sont capable d'atteindre asymptotiquement la limite de Shannon, il s'avère néanmoins que les performances du SC sur des codes de longueurs utilisées en pratique sont très insuffisantes comparées aux performances atteintes par les codes LDPC ou Turbo. D'où le développement d'une large variété d'algorithmes de décodage visant à améliorer les performances. Le but de cette section est de passer en revue les principales solutions existantes.

1.4.1 Généralités sur le décodage

Les codes polaires sont construits sur la base d'un décodage successif des bits avec la réutilisation des décisions dures précédentes. Dès lors, tout algorithme de décodage qui ne suivrait pas ce principe, sans être nécessairement inefficace, nécessiterait de revoir au moins le critère de sélection des positions d'information, mais sortirait du cadre des codes polaires. Qui plus est, il a été montré que les codes polaires ne sont pas optimaux en terme de distance minimale ou multiplicité minimale, et tout particulièrement lorsque le rendement coïncide avec celui d'un code RM. De fait, tout algorithme de décodage garantissant les performances ML gagnerait à être utilisé avec un critère de sélection des positions d'information visant à optimiser le spectre des distances, comme cela a été mis en évidence pour les algorithmes ML [6] et SD (pour Sphere Decoding, en anglais) [52, 40]. La question de la construction du code pour de tels décodeurs pour tout couple (N,K) trouve sa solution avec les codes SDO définis plus haut.

Un algorithme par propagation de croyance (communément appelé Belief Propagation en anglais), inspiré par les décodeurs LDPC, a été proposé pour les codes polaires, arguant que la nature intrinsèque-

ment séquentielle du décodage SC le rendait peu adapté pour le très haut-débit. Il repose sur des itérations droite-gauche dans le graphe polaire, sans tenir compte de l'ordonnancement spécifique du SC. De fait, ce décodeur requiert près d'une centaine d'itérations pour ne faire qu'approcher les performances du SC, et sans apporter l'avantage en terme de latence escompté [38]. Des progrès significatifs ont été obtenus récemment par un algorithme par propagation de croyance *multi-treillis* [30], exploitant différents graphes équivalents d'un même code polaire, bien que les performances restent encore assez éloignées des meilleurs décodeurs polaires.

Un autre processus est celui de l'algorithme itératif à annulation souple SCAN (pour Soft-Cancellation en anglais) [33], dans lequel chaque itération respecte l'ordonnancement du décodage SC, mais réutilise les valeurs souples des bits d'information plutôt que les décisions dures. Il a été montré que ce décodeur parvient à surpasser légèrement le SC en quelques itérations, tout en offrant une complexité radicalement inférieure à l'algorithme par propagation de croyance, tant au niveau computationnel qu'au niveau de l'utilisation de la mémoire [12]. De plus, les performances peuvent être améliorées en reconsidérant la position des bits d'information afin de réduire le nombre de mots de code de poids faible [78].

Les algorithmes les plus performants pour les codes polaires sont tous basés sur le décodage SC. Plutôt que d'explorer un unique chemin comme dans le SC, ces décodeurs s'efforcent d'explorer plusieurs chemins, en parallèle, en série, ou par un intermédiaire entre les deux. Le suite de ce chapitre est dédiée à la description de ces algorithmes.

Auparavant, il convient d'expliciter la raison fondamentale permettant à ces algorithmes de surpasser très largement le SC au point de concurrencer les codes LDPC et Turbo en longueur finie. Il a été mis en évidence le fait que la borne ML des codes polaire est en réalité relativement proche de la performance du SC, de telle sorte que même un algorithme ML n'apporterait qu'un avantage limité [95]. Pour surmonter cet obstacle, ces décodeurs utilisent le plus souvent un système concaténé d'un code polaire (interne) avec un code détecteur (externe) de type CRC (Cyclic Redundancy Check en anglais). Concrètement, un CRC est ajouté à la séquence des bits d'information $u_{\mathcal{I}}$ avant l'encodage polaire. De fait, le nombre de bits à décoder augmente de K à $K+r$, où r est le nombre de bits de CRC. Ce CRC permet d'améliorer les performances des décodeurs en servant d'indicateur de succès ou d'échec d'un décodage, avec une probabilité de non-détection d'une erreur de l'ordre de 2^{-r}. Si l'on s'en tient au code polaire $\mathcal{C}(N, K+r, \mathcal{I}')$ où $|\mathcal{I}'| = K+r$, sans tenir compte du CRC, il est possible de déterminer la distance minimale et la multiplicité, mais le CRC n'autorise en réalité qu'un nombre réduit parmi ces mots de code, de telle sorte que la multiplicité de la distance minimale, voire même la distance minimale, peuvent être améliorées dans le système concaténé.

1.4.2 Décodage par liste

Le décodage par liste des codes polaires [95] suit exactement le même ordonnancement des opérations que dans le SC, la différence reposant sur le fait qu'il explore en parallèle L chemins plutôt que l'unique chemin du SC. Par "chemin", on entend toute séquence $\hat{u}_0^{i-1}$ en cours de décodage, et telle que $\hat{u}_j = u_j$ si $j \notin \mathcal{I}$. La figure 1.9 représente l'arbre des possibilités pour un code avec $K = 4$ bits d'information. A chaque chemin est associée une métrique, calculée à partir des LLRs obtenus durant le décodage, et évaluant sa probabilité de succès. Lors du décodage d'un bit d'information $u_i, i \in \mathcal{I}$, le décodeur commence par mettre à jour la valeur de la métrique associée aux deux décisions $\hat{u}_i = 0$ et $\hat{u}_i = 1$, et ce pour chaque chemin conservés en mémoire. Pour limiter la croissance exponentielle du nombre de chemins, le décodeur ne conservera en mémoire que les L plus probables d'après la métrique, avant de continuer le décodage des bits suivants. Dans le cas d'un bit figé $u_i, i \notin \mathcal{I}$, le décodeur ne conserve que la vraie

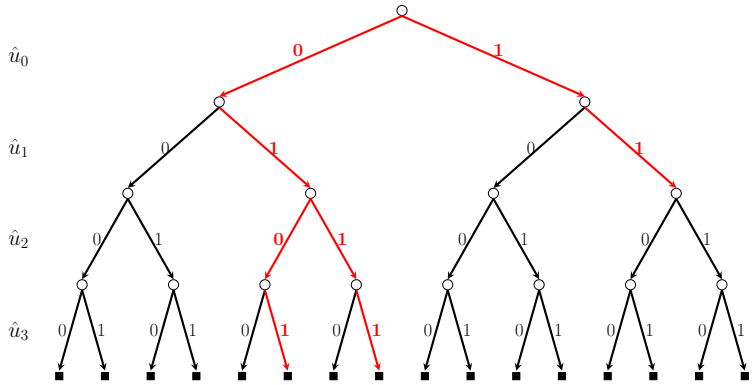


FIGURE 1.9: *Représentation en arbre d'un processus de décodage polaire : le SCL explore en parallèle L trajectoires d'après les valeurs de métriques calculées*

décision u_i et se contente uniquement de mettre à jour les valeurs des métriques d'après la valeur de L_i. A l'issue du décodage, la liste contient exactement L chemins de longueur N, soit autant de candidats pour l'estimation de $u_{\mathcal{I}}$. La sélection du candidat final est effectuée en conservant le chemin vérifiant le CRC. Si plusieurs candidats vérifient le CRC, la métrique sert à départager. Si aucun candidat ne vérifie le CRC, le résultat sera forcément en erreur.

La métrique d'un chemin $v_0^{i-1} \in \{0,1\}^i$ se calcule récursivement à partir de celle du chemin v_0^{i-2} et du LLR du bit v_{i-1} associé à cette trajectoire $L_{i-1}(y_0^{N-1}, v_0^{i-2})$ [8]:

$$\Gamma(v_0^{i-1}) = \Gamma(v_0^{i-2}) + \log\left(1 + \exp\left(-(1-2\cdot v_{i-1})\cdot L_{i-1}(y_0^{N-1}, v_0^{i-2})\right)\right), \tag{1.42}$$

avec par convention $\Gamma(\emptyset) = 0$. Une approximation importante pour les implémentations matérielles est donnée par [8]:

$$\Gamma'(v_0^{i-1}) = \begin{cases} \Gamma'(v_0^{i-2}), & \text{si } v_{i-1} = \frac{1}{2}\left(1 - \text{sign}(L_{i-1}(y_0^{N-1}, v_0^{i-2})\right) \\ \Gamma'(v_0^{i-2}) + |L_{i-1}(y_0^{N-1}, v_0^{i-2})|, & \text{sinon} \end{cases} \tag{1.43}$$

Les chemins à privilégier sont ceux minimisant cette métrique. Intuitivement, si la valeur du bit v_{i-1} est contredite par le signe de son LLR, alors le chemin sera pénalisé d'autant plus que le LLR est élevé en valeur absolue. Dans le cas contraire, aucune pénalité n'est appliquée.

A noter que ce décodeur peut aussi bien fonctionner sans CRC, auquel cas la décision finale reviendra uniquement à la métrique. Pour les distinguer, on utilisera l'acronyme CA-SCL (pour CRC-Aided SCL, en anglais) pour le décodeur avec un CRC et SCL pour celui sans. Une problématique délicate concernant ces deux décodeurs est la question du choix de $\mathcal{I}$, dans la mesure où il n'existe pas de méthode de construction analytique pour de tels décodeurs. En particulier, du fait que le SCL est capable d'approcher – sous réserve que la taille de la liste soit suffisante – la borne ML, il est possible d'améliorer les performances en utilisant un code avec une borne ML plus basse. L'intérêt d'une telle approche a été mis en évidence dans [65] où un code construit pour un SNR plus élevé que le SNR de fonctionnement est utilisé. Concernant le CA-SCL, il y a un très large consensus au sein de la communauté pour considérer la construction obtenue pour le décodeur du SC (pour $K + r$ bits sur le canal considéré) comme étant excellente, si ce n'est la meilleure. Cela peut se justifier intuitivement par le fait que l'identification du bon candidat dans la liste est déléguée au CRC, et qu'il suffit donc de chercher à maximiser le probabilité que le bon chemin soit dans la liste, la construction du SC étant des plus pertinentes pour cela.

Puisque les performances du SCL peuvent être optimisées en améliorant le spectre du code, le code SDO est un candidat qui mérite d'être considéré. La figure 1.10 présente les performances pour les

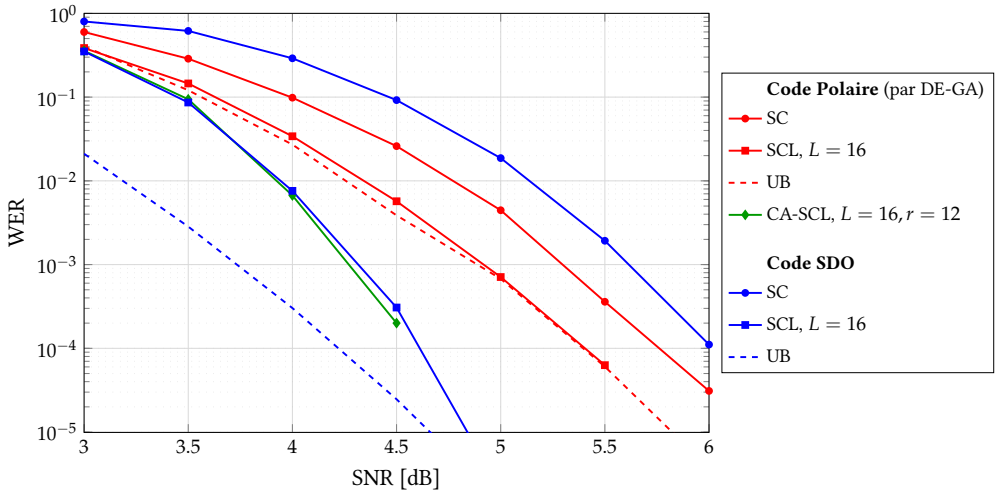


FIGURE 1.10: *Performance du code SDO et des codes polaires* $(N,K) = (512,345)$

paramètres $(N,K) = (512,345)$ d'un code polaire et du code SDO avec les décodeurs SC et SCL avec $L = 16$, ainsi que la valeur de l'UB. Les performances du CA-SCL sont également données avec $L = 16$ et $r = 12$ bits de CRC. Il peut-être observé que les performances du SCL sont bien meilleures avec le code SDO qu'avec le code polaire (puisque celui-ci bute sur la borne ML), et que le SCL du code SDO concurrence même le CA-SCL du code polaire pour une même taille de liste. Nous pensons qu'il devrait être possible d'améliorer encore les performances en trouvant le bon compromis entre le code SDO et le code polaire.

1.4.3 DÉCODAGE SÉQUENTIEL

Le décodage séquentiel fut très étudié dans les années soixante-dix pour le décodage des codes convolutifs [107]. Par rapport au décodeur par liste, le décodeur séquentiel repose sur l'exploration de chemins de longueurs variables. Il existe plusieurs approches pour la stratégie d'exploration des branches de l'arbre, chacune proposant un compromis différent entre performance, complexité calculatoire et quantité de mémoire requise, parmi lesquelles l'algorithme de Fano, l'algorithme Stack ou l'algorithme de Creeper [51]. Pour les codes polaires, l'application d'un tel décodeur a été proposé tantôt sous le nom de SC-Stack (SCS) [72, 23] tantôt de "décodeur séquentiel"[63, 101, 100], mais les deux utilisent en réalité la stratégie d'exploration de l'algorithme de Stack, et ne diffèrent que par la définition de la métrique.

Par rapport au décodeur par liste, l'idée fondamentale du décodeur séquentiel est d'explorer des chemins de longueurs différentes, conservés non plus dans une liste mais dans une "pile" (traduction de stack en anglais) contenant les D meilleurs chemins en cours d'exploration. Le principe est de ne prolonger à chaque étape du processus que le chemin le plus probable d'après les valeurs des métriques. Lors du prolongement d'un chemin par un bit d'information, les deux décisions sont envisagées et les chemins associés sont stockés dans la pile. Seuls les D chemins les plus probables sont conservés dans la pile, les autres étant simplement éliminés. En l'absence de CRC, le premier chemin atteignant la longueur N met fin au décodage et constitue le candidat final. Avec un CRC, le décodage est interrompu lorsqu'un chemin vérifie le CRC ou qu'un total de D chemins de longueur N a été obtenu.

Le principal défaut de ce décodeur est qu'il requière une quantité de mémoire significativement plus élevée que le décodage par liste, sous peine de détérioration dramatique des performances [23], et hypothéquant sérieusement la possibilité qu'une implémentation matérielle atteigne une latence très faible. Une autre problématique de cet algorithme est la définition de la métrique permettant de comparer efficacement des chemins de longueurs différentes. La métrique proposée dans [72] est équivalente à celle du

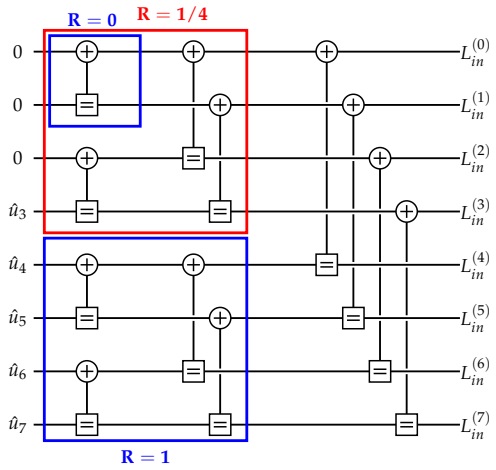


FIGURE 1.11: *Décodage SC simplifié et SC simplifié rapide pour un code $N = 8$*

décodeur par liste, mais sans prise en compte des LLRs des bits figés. Dans [100], il est proposé d'utiliser la métrique du décodage par liste (en tenant compte des bits figés), mais d'ajouter un biais pour affiner la comparaison des chemins de longueurs différentes. Ce biais est déterminé préalablement au décodage à partir des fiabilités des canaux virtuels.

1.4.4 DÉCODAGE À INVERSION

Le décodage à inversion [3] consiste à effectuer des tentatives de décodages les unes après les autres tant qu'aucune ne passe le CRC ni qu'un nombre maximal de tentatives a été atteint. Dans [3], les tentatives sont obtenues en inversant une unique décision par rapport au décodage SC. La position des inversions est déterminée par les T indices ayant les valeurs de LLR les plus faibles lors du décodage SC. Le concept de décodage à inversion est le sujet du chapitre 3, nous n'en dirons donc pas plus pour l'instant.

1.4.5 DÉCODEURS FAIBLE LATENCE

Puisque le processus de décodage SC est de nature intrinsèquement séquentielle, il est difficilement compatible avec les applications requérant une très faible latence. Cependant, un certain nombre d'optimisations algorithmiques peuvent être implémentées de manière à réduire simultanément la complexité et la latence sans affecter les performances. Cette version faible latence du SC, appelée décodeur SC simplifié (SSC) [4], permet la réduction significative du nombre de calculs de LLRs, en exploitant les propriétés de l'ensemble $\mathcal{I}$ du code. Le principe est d'identifier dans le graphe des blocs de bits d'information (appelés aussi nœuds) dont le décodage peut être accéléré. Dans le SSC, ceux de rendement $R = 0$ (contenant uniquement des bits figés) ne sont pas décodés, tandis que ceux de rendement $R = 1$ (uniquement des bits d'information) sont décodés sous forme de décisions dures. Un exemple est fourni dans la figure 1.11 pour un code $(N,K) = (8,5)$, où blocs de rendement $R = 1$ et $R = 0$ sont visualisés en bleu. Un degré supplémentaire d'optimisation est obtenu par le décodage SSC rapide (Fast SSC) dans lequel le décodage de certains nœuds de rendement $R \in]0,1[$ est accéléré [84]. Par exemple, un nœuds de rendement $R = 1/N'$, comme celui en rouge sur la figure, est en fait un code à répétition qui peut se décoder en un seul cycle d'horloge. Pour le code $(N,K) = (8,5)$ pris en exemple, le nombre de calculs de LLRs vaut 24, 11 et 9 tandis que le nombre de cycles d'horloge vaut 14, 4 et 3 pour le SC, SSC et Fast-SSC respectivement.

Il a été mis en évidence que le décodage par liste peut également bénéficier d'optimisations similaires [41]. Pour les nœuds de rendement $R = 0$, la connaissance des LLRs à l'entrée du bloc est suffisante pour mettre à jour les valeurs des métriques sans altérer les performances [83]. Pour les nœuds de rendement

$R = 1$, le décodeur ne procède plus à un décodage SC, mais utilise le décodage de Chase, en recherchant les candidats les plus probables à partir des LLRs d'entrée du bloc [83]. Pour les autres patterns pouvant bénéficier d'optimisations, le lecteur pourra se référer à [83] et aux travaux concomitants.

Dans [37], l'application de ces mêmes optimisations algorithmiques est discutée pour un décodeur à inversion. Il est proposé que les LLRs des nœuds de rendement $R = 1$ ne soient décodés par un processus SC qu'à la première tentative, afin de calculer les métriques nécessaires à la poursuite du décodage, mais que ceux-ci soient par suite décodés en décisions dures. La question de certains nœuds de rendement $R \in]0,1[$ est également étudiée.

Finalement, il semble qu'il n'existe pas d'article de recherche concernant un décodeur SCS simplifié, mais il est clair que celui-ci peut également bénéficier des mêmes optimisations que le décodeur par liste.

1.5 Les codes polaires dans la 5G

Dans la cinquième génération de standards pour la téléphonie mobile (5G), les données utilisateurs ainsi que les informations de contrôle de/vers la couche MAC sont codés/décodés pour assurer la transmission sur le lien radio. Le processus complet d'encodage consiste en une combinaison de techniques de détection d'erreurs (ajout de bits de CRC), correction d'erreurs (utilisation d'un code correcteur), ajustement du rendement de codage (rate matching, en anglais), et d'entrelacement.

La correspondance entre le canal de transport, le canal physique et le type de code correcteur associé est donnée dans le tableau 1.2. Il peut être observé que les codes Polaires ont été adoptés pour les canaux de contrôle, à savoir le canal UCI (pour Uplink Control Information, en anglais) en liaison montante, et les canaux BCH (pour Broadcast Channel, en anglais) et DCI (pour Downlink Control Information, en anglais) en liaison descendante.

Table 1.2: *Correspondance canal de transport, canal physique et codage dans la norme 5G*

Canal de transport	Canal Physique	Codage canal
Liaison montante (Uplink)		
Random Access Channel (RACH)	PRACH	séquence de Zadoff-Chu
Uplink shared channel (UL-SCH)	PUSCH	LDPC
Uplink Control Information (UCI)	PUCCH, PUSCH	Polaire, ou codes en bloc courts
Liaison descendante (Downlink)		
Broadcast Channel (BCH)	PBCH	Polaire
Downlink shared channel (DL-SCH)	PDSCH	LDPC
Paging Channel (PCH)	PDSCH	LDPC
Downlink Control Information (DCI)	PDCCH	Polaire

Le tableau 1.3 fournit les valeurs des paramètres des codes pour chacun des canaux physiques où les codes polaires ont été adoptés (selon la norme 3GPP TS 38.212 V15.3.0 de septembre 2018). Le processus complet d'encodage d'un code polaire dans le 5G se décompose en 8 étapes (les notations introduites ici sont celles utilisées dans la norme [1], et non pas dans le reste du manuscrit):

0. Point de départ : A bits d'information et E bits transmis[5]
1. Ajout de L bits de CRC aux A bits d'information, de manière à obtenir les $K = A + L$ bits à encoder
2. Calcul de la longueur $N = 2^n$ du code polaire : un encadrement $n_{min} \leq n \leq n_{max}$ est imposé en fonction du canal physique (cf tableau 1.3), mais la valeur exacte de n est déterminée à partir de

[5]Le paramètre A est le nombre de bits d'information par bloc de code, à l'issue la procédure de segmentation, définie dans 5.2.1 de [1] et non détaillée ici.

E. En général, $n = \lceil \log_2(K) \rceil$, mais des contraintes supplémentaires sont appliquées pour garantir $K/N \geq 1/8$ ou éviter de devoir poinçonner/raccourcir une fraction trop importante des bits à l'étape 7.

3. Pré-entrelacement : un entrelaceur est appliqué avant encodage, décrit dans la Table 5.3.1.1-1 de [1]. Son intérêt est essentiellement lié à la détection aveugle sur le canal PBCH [14]. L'application ou non de cet entrelaceur est indiquée par le booléen $I_{il} \in \{0,1\}$.
4. Positionnement des bits d'information : une unique séquence a été choisie pour déterminer les positions d'information et les positions figées. Cette séquence est décrite dans la Table 5.3.1.2-1 de [1], listant les indices des canaux virtuels par fiabilité croissante pour une longueur $N_{max} = 1024$. A partir de cette séquence, le positionnement des bits d'information s'obtient comme suit:
   a Déterminer l'ensemble des indices des bits figés, noté Q_F^N. Cela tient compte de l'ajustement du rendement à l'étape 8, ainsi que des fiabilités des canaux virtuels.
   b L'ensemble des indices des bits non-figés est obtenu par $Q_I^N = \{0,\dots,N-1\} \backslash Q_F^N$, avec $|Q_I^N| = K + n_{pc}$. Cela regroupe à la fois les indices des bits d'information et des éventuels bits de parités introduits pour aider le décodage. Le positionnement des deux catégories de bits suit la règle suivante dans l'ordre:
      - $(n_{pc} - n_{pc}^{wm})$ bits de parités sont positionnés sur les indices les moins fiables de Q_I^N
      - n_{pc}^{wm} bits de parités sont positionnés sur les indices de Q_I^N restants, dont les lignes de G_N associées sont de poids les plus faibles
      - les K bits d'information sont positionnés sur les indices restants
5. Encodage polaire : $x_0^{N-1} = u_0^{N-1} \cdot G_N$
6. Appliquer l'entrelacement par bloc : la séquence codée est subdivisée en sous-blocs de longueurs $N' = 32$ bits, chacun étant entrelacée par une même permutation définie dans la Table 5.4.1.1 de [1].
7. Ajustement du rendement de codage (rate-matching) : utilisation du poinçonnage, du raccourcissement ou de la répétition (en fonction du rendement) afin d'obtenir exactement les E bits à transmettre. Lorsque $N > E$, Le poinçonnage est privilégié pour les rendements $\frac{A+L}{E} \leq \frac{7}{16}$ et le raccourcissement dans le cas contraire.
8. Post-entrelaceur : un entrelacement est finalement appliqué afin d'optimiser les performances pour les modulations d'ordre supérieur. L'application ou non de cet entrelaceur est indiqué par le booléen $I_{bil} \in \{0,1\}$.

Les modulation supportées dans la 5G sont BPSK, QPSK, 16-QAM, 64-QAM et 256-QAM. Pour plus d'informations concernant les code polaires dans les standards 5G, le lecteur pourra se référer directement à la norme [1] ou au précieux travail de vulgarisation [14].

TABLE 1.3: *Specifications 5G pour les codes polaires*

	Liaison montante • Uplink control information (PUCCH/PUSCH)			**Liaison descendante** • Broadcast channel (PBCH) • Downlink Control information (PDCCH)
	$A > 19$	$12 \leq A \leq 19$		
		$E - A \leq 175$	$E - A > 175$	
$N = 2^n$	$5 \leq n \leq 10$			$5 \leq n \leq 9$
L	11	6		24
n_{pc}^{wm}	0	3		0
n_{pc}	0	0	1	0
I_{il}	0			1
I_{bil}	1			0

1.6 CONCLUSION

Ce chapitre a permis d'expliciter les notions fondamentales relatives aux codes polaires. Ceux-ci exploitent le phénomène de polarisation garantissant asymptotiquement d'atteindre la limite de Shannon pour tout canal binaire discret sans mémoire, en utilisant les canaux virtuels les plus fiables pour transmettre l'information. Si la position des bits d'information dépend du canal utilisé, la régularité de la transformation récursive employée impose toutefois un ordre partiel universel entre certains canaux virtuels. Cet ordre partiel a pour conséquence de conférer aux codes polaires des propriétés algébriques singulières permettant notamment de déterminer l'ensemble $\mathcal{I}$ correspondant au code respectant l'ordre partiel maximisant la distance minimale et minimisant le nombre de mots de code de poids faible. Dans le même temps, ce chapitre a permis de revenir sur les principaux algorithmes de décodage pour les codes polaires. Le décodeur SC est asymptotiquement optimal, de faible complexité, mais reste insuffisant en longueur finie. Les décodeurs les plus performants sont tous basés sur le décodeur SC, mais se permettent d'explorer plusieurs chemins. Trois grandes stratégies d'explorations des chemins existent, appliquées dans les algorithmes par liste, séquentiels et à inversion, offrant chacune un compromis entre performance et complexité diffèrent.

2 Poinçonnage et raccourcissement des codes polaires

Contenu du Chapitre

Les codes polaires considérés dans le chapitre précédent ont une longueur nécessairement égale à une puissance de deux. En pratique cependant, des codes flexibles en longueur sont requis. Les deux principales méthodes pour cela sont appelées le poinçonnage et le raccourcissement d'un code. Une approche alternative est d'utiliser un code polaire construit à partir d'un ou plusieurs autres matrices noyaux. Ce chapitre a pour but de revenir sur ces différentes techniques, et de montrer dans quelle mesure ces méthodes sont reliées.

2.1 Codes polaires flexibles

2.1.1 Poinçonnage

Un code polaire *poinçonné* $\mathcal{C}_p(N-N_p,K,\mathcal{I})$ est obtenu à partir d'un code polaire *mère* $\mathcal{C}(N,K,\mathcal{I})$ en ne transmettant que $N-N_p$ bits codés. Les bits non transmis sont indiqués par le *motif de poinçonnage*, noté $P \in \{0,1\}^N$. Puisque le décodeur ne dispose pas d'estimations des bits codés x_i lorsque $P(i)=1$, celui-ci sera initialisé avec $L_{in}(i)=0$ lorsque $P(i)=1$ et au LLR calculé à partir de la valeur reçue y_i lorsque $P(i)=0$. On notera par $y[P]_i=0$ si $P(i)=1$, tandis que $y[P]_i=y_i$ sinon.

Un code poinçonné transmis sur un canal W peut s'interpréter comme un code non-poinçonné transmis sur des canaux parallèles $\{W_i\}_{i\in\{0,\dots,N-1\}}$ où W_i est soit le canal W soit un canal totalement bruité ($I(W_i)=0$). Considérons un noyau G_2 dont les bits codés sont transmis sur deux canaux W_0 et W_1 de capacités symétriques $I(W_0)$ et $I(W_1)$ respectivement. La généralisation des équations de polarisation relatives à ce noyau s'écrit [22]:

$$I(W^+)+I(W^-)=I(W_0)+I(W_1) \tag{2.1}$$
$$\min(I(W_0),I(W_1)) \geq I(W^-) \tag{2.2}$$
$$\max(I(W_0),I(W_1)) \leq I(W^+) \tag{2.3}$$

avec égalité si et seulement si $I(W_0)\in\{0,1\}$ ou $I(W_1)\in\{0,1\}$. En particulier:

$$I(W^-)=0 \Leftrightarrow I(W_0)=0 \text{ ou } I(W_1)=0$$
$$I(W^+)=0 \Leftrightarrow I(W_0)=0 \text{ et } I(W_1)=0$$

Si la polarisation des canaux virtuels est maintenue pour un code poinçonné, il apparaît cependant que, par induction, le fait que $I(W_0)=0$ implique qu'un ou plusieurs canaux virtuels peuvent être de capacité symétrique rigoureusement nulle. Dans la suite, on notera $W[P]_N^{(i)}$ les canaux virtuels du code poinçonné.

Définition 2 *Soit un code polaire $\mathcal{C}(N-N_p,K,\mathcal{I})$ poinçonné par le motif P. On appelle motif d'effacement le vecteur binaire $E[P]\in\{0,1\}^N$ tel que:*

$$E[P](i)=1 \iff I(W[P]_N^{(i)})=0 \tag{2.4}$$

La proposition suivante a été démontrée dans [91]:

Proposition 6 *Soit un code polaire $\mathcal{C}(N-N_p,K,\mathcal{I})$ poinçonné par le motif P, alors:*

$$|E[P]|=|P|=N_p, \tag{2.5}$$

Puisque les positions des bits d'information d'un code polaire correspondent aux canaux virtuels les plus fiables, il est clair que:

$$\mathcal{I} \cap \text{Supp}(E[P]) = \emptyset, \tag{2.6}$$

où $\text{Supp}(E[P])$ correspond au support de $E[P]$, *i.e.* $\text{Supp}(E[P]) = \{i \mid E[P](i) = 1\}$.

Tout comme pour un code non-poinçonné, les canaux virtuels d'un code poinçonné peuvent être partiellement ordonnés, d'après la règle donnée par l'équation (1.15):

$$b_n^{(i)}[k] \geq b_n^{(j)}[k] \; \forall k \in \{0,\dots,n-1\} \Rightarrow Z(W[P]_N^{(i)}) \leq Z(W[P]_N^{(j)}) \; \forall W \tag{2.7}$$

En revanche, la règle d'ordre partiel (1.17) ne s'applique plus pour tout motif de poinçonnage P. Un contre-example simple est donné par le motif de poinçonnage $P = [1,0,1,0]$, où l'on vérifie que $I(W_4^{(2)}) = 0 < I(W_4^{(1)})$.

Le décodage d'un code polaire poinçonné est parfaitement similaire au décodage du code mère, de sorte que tous les algorithmes de décodages présentés dans le Chapitre 1 peuvent être appliqués.

La problématique majeure relative à un code poinçonné est le choix du motif de poinçonnage de manière à garantir les meilleures performances du code poinçonné $\mathcal{C}(N - N_p, K, \mathcal{I})$ pour le décodage SC. Cette difficulté est accentuée par le fait que le choix optimal des positions d'information $\mathcal{I}$ d'un code polaire est dépendant du canal – et donc d'un éventuel motif de poinçonnage P. Il est clair que $Z(W[P]_N^{(i)}) \geq Z(W_N^{(i)})$, mais il convient de noter que la dégradation $Z(W_N^{(i)}) - Z(W[P]_N^{(i)})$ n'est pas uniforme pour tout indice $i \in \{0,\dots,N-1\}$ de sorte que l'ensemble $\mathcal{I}$ obtenu sans poinçonnage n'est en général pas lié à celui obtenu lorsque le poinçonnage est pris en compte. Une difficulté supplémentaire est que le meilleur motif P pour un code de rendement $K/(N - N_p)$ n'est pas garanti de rester optimal pour une autre valeur de rendement. Pour un motif de poinçonnage P donné, il est néanmoins possible de réactualiser les méthodes de construction de la section 1.3.2 afin de trouver le meilleur ensemble $\mathcal{I}$. Par contre, trouver le couple $(P, \mathcal{I})$, où P est un motif de poinçonnage avec $|P| = N_p$, et $\mathcal{I}$ indique les indices des bits d'information, de manière à minimiser le taux d'erreur du code $\mathcal{C}(N - N_p, K, \mathcal{I})$ pour le SC est excessivement complexe. Un cas légèrement plus simple, intervenant dans les systèmes utilisant de la retransmission de type HARQ (Hybrid Automatic Repeat Request, en anglais), est le cas où l'ensemble $\mathcal{I}$ est imposé et où il s'agit de trouver le motif de poinçonnage P minimisant le taux d'erreur du SC.

Dans la littérature, on peut distinguer trois grandes approches pour optimiser le motif de poinçonnage. La première exploite les techniques d'évolution de densité pour étudier les performances des motifs de poinçonnage tant que la complexité reste abordable. Pour réduire la complexité par rapport à l'exploration exhaustive, des motifs sont prouvés comme étant équivalents et sont écartés de l'analyse [54, 109]. L'autre approche consiste à concevoir un code poinçonné comme un code utilisant un ou plusieurs noyaux et en s'efforçant de rechercher les plus pertinents d'entre eux d'après l'*exposant* (cf plus loin) des matrices [91]. Finalement, la dernière approche est plus pragmatique dans la mesure où elle n'implique pas un processus d'optimisation complexe, mais fournit des solutions simples ayant fait leurs preuves en pratiques. La première est appelée le motif QUP (pour Quasi-Uniform Puncturing, en anglais) et qui consiste à poinçonner les bits dans l'ordre $P = [1,\dots,1,0,\dots,0]$ [73]. Il a également été proposé d'utiliser la permutation à inversion de bit du motif QUP [15]. Finalement, une dernière possibilité est de choisir les positons poinçonnées d'après les positions des canaux virtuels du code mère les moins fiables [109].

Précisons que la technique de poinçonnage proposée dans la 5G ne s'apparente à aucune méthode connue. En effet, la norme prévoit de poinçonner les positions en partant du début (comme pour QUP) mais uniquement après l'application de l'entrelaceur par bloc (cf section 1.5). Si cela peut être équivalent au motif QUP pour certaines valeurs de N_p, cela n'est pas vrai dans le cas général.

2.1.2 Raccourcissement

Le raccourcissement d'un code polaire mère $\mathcal{C}(N,K)$ a en commun avec le poinçonnage de ne transmettre que $N - N_s$ bits codés, mais cette fois-ci il est fait en sorte que les valeurs de ces bits non-transmis soient connues au niveau du décodeur. On notera par S le motif de raccourcissement, avec $y[S]_i = (1-2x_i)\infty$ si $S(i) = 1$, tandis que $y[S]_i = y_i$ sinon.

Similairement à un code poinçonné, un code raccourci peut s'interpréter comme un code transmis sur des canaux parallèles avec soit $W_i = W$ soit W_i est un canal parfait. Les équations de polarisation (2.3) donnent:

$$I(W^-) = 1 \Leftrightarrow I(W_0) = 1 \text{ et } I(W_1) = 1$$
$$I(W^+) = 1 \Leftrightarrow I(W_0) = 1 \text{ ou } I(W_1) = 1$$

Cette fois-ci, certains canaux virtuels deviennent parfaits, *i.e.* $I(W_N^{(i)}) = 1$. Cependant, ces canaux parfaits sont en réalité inutilisables, puisque les valeurs des bits correspondants doivent être fixées de manière à garantir que les bits non-transmis soient connus. On définit le motif $E[S]$ associé au motif de raccourcissement S par:

$$E[S] = \{i \in \{0,\dots,N-1\}, I(W_N^{(i)}) = 1\} \tag{2.8}$$

Il s'ensuit que $\mathcal{I} \cap \text{Supp}(E[S]) = \emptyset$.

Dans [104], une technique de raccourcissement (bien que présentée comme du poinçonnage) est proposée, et constitue l'exact pendant du motif QUP pour le poinçonnage, à savoir que les bits raccourcis sont choisis dans l'ordre en partant de la fin: $S = [0,\dots,0,1,\dots,1]$. Dans la suite, ce motif sera dénommé QUS (pour Quasi-Uniform Shortening en anglais), du fait de la similarité avec l'approche QUP pour le poinçonnage. La permutation à inversion de bit de ce motif est considérée dans [15]. Une autre solution consiste à fixer à 0 les positions correspondant aux canaux virtuels les plus fiables du code mère. Dans [62], une méthode pour explorer l'ensemble des motifs non-équivalents est proposée pour les codes courts. Pour les codes plus longs, il est proposé de ne considérer le raccourcissement que d'un sous-ensemble de positions, ou bien de répéter le motif de raccourcissement de manière cyclique lorsque $N_p = 2^k \cdot N'_p$.

Tout comme pour le cas du poinçonnage, l'entrelaceur par bloc empêche d'assimiler la solution considérée dans la 5G à une solution connue. Précisions que le choix entre le poinçonnage ou le raccourcissement dans la 5G est fixée par le rendement du code. Pour les rendements inférieurs à $\frac{7}{16}$ le poinçonnage est privilégié tandis que le raccourcissement est utilisé dans le cas contraire, conformément aux observations présentées dans [15].

2.1.3 Utilisation d'autres noyaux

L'alternative aux deux techniques présentées ci-dessus est d'utiliser une autre matrice noyau, ou une combinaison de plusieurs noyaux. Il a été montré que le phénomène de polarisation se manifeste également en appliquant la puissance de kronecker sur toute matrice F_ℓ carrée, inversible et non triangulaire supérieure après toute permutation de colonnes [55]. Ce résultat a été étendu dans le cas où F_ℓ s'exprime comme le produit de kronecker de matrices respectant les conditions précédentes, $F_\ell = F_{\ell_1} \otimes \cdots \otimes F_{\ell_m}$ [58, 11]. Dans ce dernier cas, la taille du noyau est donnée par $\ell_1 \cdot \ell_2 \cdots \ell_m$, où $\ell_1,\dots,\ell_m$ sont les tailles des noyaux, de sorte qu'il est possible d'obtenir une large variété de longueurs de code.

Il convient de préciser que l'étude des structures basées sur d'autres noyaux que celui proposé par Arikan n'a pas, en premier lieu, été motivée par la flexibilité sur la longueur du code obtenue, mais bien par les attentes relatives à l'amélioration de la vitesse de convergence du phénomène de polarisation.

Pour cela, deux grandeurs théoriques d'un noyau, noté simplement F dans la suite, de taille $\ell \times \ell$, ont été définies:

- l'*exposant d'erreur* $\mathbf{E}(F)$ [55]. Pour tout noyau F de taille $\ell \times \ell$, considérons un canal (BDMC) W et un code polaire de longueur $N = \ell^n$ et de rendement donné $R < I(W)$. Alors, pour tout $\beta < \mathbb{E}(F)$, si n est suffisamment large, la probabilité d'erreur par bloc du décodage SC, notée P_e, est majorée par:
$$P_e \leq 2^{-N^\beta} \tag{2.9}$$
- l'*exposant d'échelle* $\mu(F)$. Pour tout noyau F de dimension $\ell \times \ell$, considérons un canal (BDMC) W et un code polaire de rendement $R < I(W)$ et une probabilité d'erreur P_e. Alors, la longueur du code nécessaire pour garantir que la probabilité d'erreur par bloc du SC soit inférieure à P_e vérifie:
$$N \leq \frac{\alpha}{(I(W) - R)^{\mu(F)}}, \tag{2.10}$$
où α est une constante positive dépendante de P_e. [42, 67].

Ces deux grandeurs constituent les critères privilégiés pour choisir les meilleures matrices noyaux ([55] pour l'exposant d'erreur, et [34] pour l'exposant d'échelle). Précisons toutefois que l'exposant d'échelle est nettement plus complexe à calculer (les méthodes proposées considèrent un canal BEC et nécessite un processus heuristique), de sorte que l'exposant d'erreur est largement plus usuel dans la littérature.

Pour toute matrice F de dimension $\ell \times \ell$, l'exposant d'erreur peut se calculer comme suit [55]:

$$\mathbf{E}(F) = \frac{1}{\ell} \sum_{i=0}^{\ell} \log_\ell(D_i), \tag{2.11}$$

où les $\{D_i\}$ sont appelées les distances partielles et se calculent à partir des lignes de la matrices F, notées $F^{(i)}$ d'après:

$$D_i = d_h(F^{(i)}, \langle F^{(i+1)}, \ldots, F^{(\ell)} \rangle), \tag{2.12}$$

où $\langle F^{(i+1)}, \ldots, F^{(\ell)} \rangle$ est l'espace vectoriel engendré par les vecteurs binaires $F^{(i+1)}, F^{(i+2)}, \ldots, F^{(\ell)}$ et d_h est la distance de hamming entre un vecteur et un espace vectoriel. Pour la dernière ligne, la distance partielle est donnée par $D_{N-1} = |F^{(N-1)}|$.

Précisons tout de suite que l'exposant d'erreur vérifie $\mathbf{E}(F^{\otimes n}) = \mathbf{E}(F)$ [55] et qu'appliquer les opérations suivantes sur une matrice noyau n'affecte pas la valeur de celui-ci:

- permutations de colonnes
- remplacement d'une ligne $F^{(i)}$ par $F^{(i)} \oplus F^{(j)}$ avec $j > i$

Il a été montré que ce résultat est également valide pour l'exposant d'échelle [34]. On notera que, à partir de ces deux opérations, toute matrice peut être transformée en une matrice triangulaire inférieure (éventuellement l'identité si la matrice initiale ne polarise pas). Cependant, cette matrice n'est pas unique. Par exemple, les noyaux $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ et $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ s'obtiennent l'un l'autre en permutant les deux premières colonnes puis en remplaçant la première ligne par $F^{(0)} \oplus F^{(1)}$.

Dans [55], il est prouvé que $\mathbf{E}(G_2) = \frac{1}{2}$ et qu'aucune matrice avec $\ell \leq 15$ n'a d'exposant supérieur. Finalement, une méthode est proposée, basée sur le raccourcissement de codes BCH, pour trouver des matrices d'exposant supérieur à $\frac{1}{2}$ à partir de $\ell = 16$.

Dans [58, 11], il est montré que l'exposant d'erreur d'une structure multi-noyaux est donné par :

$$\mathbf{E}(F_{\ell_1} \otimes F_{\ell_2} \otimes \cdots \otimes F_{\ell_m}) = \sum_{i=1}^{m} \frac{\mathbf{E}(F_{\ell_i})}{\log_{\ell_i}(\ell_1 \ell_2 \cdots \ell_m)}, \tag{2.13}$$

où ℓ_i est la dimension du noyaux A_i.

Si les bonnes performances des structures multi-noyaux ont été confirmées notamment dans [44, 26], le décodage devient en contrepartie plus complexe, de l'ordre de $\mathcal{O}(\ell^2 N \log(N))$ [44]. En effet, la méthode généralement appliquée est de déterminer les équations de mise-à-jour des messages entre l'entrée et la sortie d'un noyau, sans gestion efficace des LLRs intermédiaires.

Malgré tout, la notion d'exposant constitue un outil puissant pour discriminer les matrices et sélectionner les plus prometteuses, et il a été proposé d'utiliser ce critère pour optimiser les motifs de poinçonnage dans [91].

2.1.4 Contributions

Les contributions de ce chapitre à la problématique de la flexibilité des codes polaires sont les suivantes. Tout d'abord, partant de la structure spécifique des codes polaires, une classification précise des motifs est introduite, permettant d'écarter un grand nombre de motifs équivalents. Si le passage en revue des motifs non-équivalents a été exploré de manière indépendante dans [54] pour le poinçonnage, ou dans [62] pour le raccourcissement, notre contribution est d'introduire deux sous catégories appelées motifs symétriques et non-symétriques, bien caractérisées à partir des lignes de la matrice G_N. Cette catégorisation est notamment utilisée pour déterminer un critère permettant de se concentrer sur un nombre réduit de motifs tous non-équivalents, afin de simplifier la recherche d'un bon motif pour des longueurs de codes ne permettant pas une analyse exhaustive. De plus, la caractérisation proposée permet de rendre explicite le lien entre l'exploration des motifs de poinçonnage et de raccourcissement, de sorte que les deux peuvent être traités conjointement. Finalement, les propriétés des motifs symétriques permettent également de renforcer le lien entre un code poinçonné ou raccourci et un code basé sur une structure multi-noyaux.

2.2 Classification des motifs de poinçonnage et de raccourcissement

2.2.1 Notations

Les notations et terminologies suivantes seront utilisées dans la suite de ce chapitre:

- Pour tout vecteur binaire $A \in \{0,1\}^N$, $|A|$ est le nombre de "1" dans A, *i.e.*:

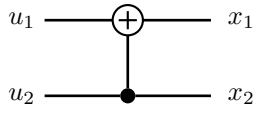
$$|A| = \sum_{i=0}^{N} A_i$$

- $A = \bigvee_{\ell=0}^{L-1} A_\ell$ représente l'opération de "ou inclusif" appliquée bit à bit entre les vecteurs binaires $A_0, A_1, \ldots, A_{L-1}$. Ainsi:

$$A(i) = 1 \Leftrightarrow \exists \ell \in \{0, \ldots, L-1\}, A_\ell(i) = 1$$

  On dira dans ce cas que A est la *réunion* des vecteurs $A_0, A_1, \ldots, A_{L-1}$ (en réalité c'est le support de A – *i.e.* l'ensemble des positions i, telles que $A(i) = 1$ – qui est la réunion des supports des vecteurs $A_0, \ldots, A_{L-1}$).

- Pour deux vecteurs binaires $(A, B) \in (\{0,1\}^N)^2$, on dira que A est *inclus* dans B, noté $A \subseteq B$, si $A \vee B = B$ (Là encore, c'est le support de A qui est effectivement inclus dans le support de B).



(a) Graphe du noyau

motif de poinçonnage	[0,0]	[1,0]	[0,1]	[1,1]
motif d' effacement	[0,0]	[1,0]	[1,0]	[1,1]

(b) Liste des motifs

FIGURE 2.1: *Liste des motifs pour le noyau G_2 : les motifs d'effacements possibles sont soit le motif $[0,0]$ soit exactement une ligne de G_2.*

- $A <_{lex} B$ représente l'ordre lexicographique de A et B, *i.e.*, si l'on suppose $A \neq B$:

$$A <_{lex} B \Leftrightarrow A(k) < B(k) \text{ où } k = \min\{i | A(i) \neq B(i)\}$$

- La permutation à inversion de bit est notée B_N

2.2.2 Caractérisation des motifs d'effacement

Rappelons qu'un code poinçonné de N_p positions donne lieu à exactement N_p canaux virtuels de capacités symétriques nulles, dont les positions sont désignées par le motif d'effacement. Ce motif d'effacement est unique et indépendant du canal W. Nous proposons la caractérisation suivante des motifs d'effacements utilisant les lignes de la matrice G_N:

Proposition 7 *Si $E[P]$ est le motif d'effacement d'un motif de poinçonnage P, alors $E[P] = \mathbf{0}_N$ (si $P = \mathbf{0}_N$) ou $E[P]$ est une réunion de lignes de la matrice G_N:*

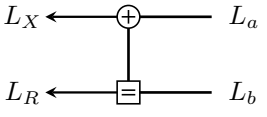
$$\exists \mathcal{L} \subset \{0,\dots,N-1\}, E[P] = \bigvee_{i \in \mathcal{L}} G_N^{(i)}$$

La démonstration est donnée en Annexe B.1. Une illustration au niveau du noyau est donnée dans la figure 2.1. Il sera montré plus loin que cette propriété est en réalité une équivalence, en fournissant une règle simple pour trouver, pour tout vecteur E correspondant à une réunion de lignes de G_N, un motif de poinçonnage ayant exactement E pour motif d'effacement (*i.e.* tel que $E = E[P]$).

2.2.3 Permutations élémentaires et motifs équivalents

Des motifs de poinçonnage distincts peuvent avoir les exactement les mêmes performances, de sorte qu'il est suffisant de se concentrer sur un sous-ensemble réduit de motifs par rapport aux $\binom{N}{N_p}$ motifs distincts de longueur N et de poids $|P| = N_p$ [54]. Considérons $u_0^{N-1} = \mathbf{0}_N$ et un bloc noyau quelconque du graphe polaire, d'entrées L_a et L_b, de sorties L_X et L_R pour les nœuds xor et répétition respectivement (dans le sens du décodage). Si l'on considère un décodage genie-aided, alors les équations de mise à jour sont données par:

$$L_X = 2 \cdot \operatorname{atanh}\left(\tanh\left(\frac{L_a}{2}\right) \cdot \tanh\left(\frac{L_b}{2}\right)\right)$$
$$L_R = L_a + L_b$$



Il peut être observé que les deux entrées L_a et L_b jouent des rôles symétriques, de sorte que permuter les deux entrées du noyau n'affecte pas les sorties L_X et L_R. Plus généralement, il est possible de définir

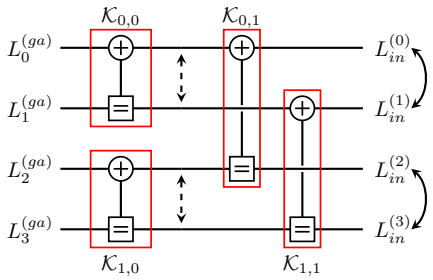


FIGURE 2.2: *Permutations élémentaires pour* $N = 4$

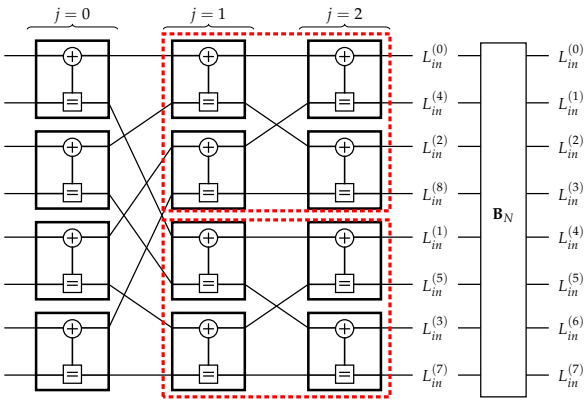


FIGURE 2.3: *Représentation équivalente du graphe polaire pour* $N = 8$

un ensemble de permutations à appliquer sur les LLRs d'entrée du décodeur, de manière à ce que les LLRs genie-aided en sortie du décodeur ne soient pas modifiés, car celles-ci se contentent d'inverser les entrées d'un ou plusieurs noyaux du graphe polaire. Un exemple pour le cas $N = 4$ est fourni dans la figure 2.2. Hormis les deux permutations consistant à inverser les entrées des noyaux $\mathcal{K}_{0,1}$ et $\mathcal{K}_{1,1}$, définies par $\phi_{0,1} : [0,1,2,3] \to [2,1,0,3]$ et $\phi_{1,1} : [0,1,2,3] \to [0,3,2,1]$ respectivement, il existe une permutation supplémentaire, décrite par les flèches de la figure, et définie par $\phi_{0,0} : [0,1,2,3] \to [1,0,3,2]$. Cette permutation revient en fait à permuter à la fois les entrées des noyaux $\mathcal{K}_{0,0}$ et $\mathcal{K}_{1,0}$, et n'affecte donc pas les LLRs genie-aided.

Afin de visualiser ces permutations dans le cas général, il est utile de considérer une représentation équivalente du graphe polaire, visible sur la figure 2.3, où, après avoir appliqué la permutation à inversion de bit (notée B_N) sur les LLRs d'entrée, les blocs noyaux apparaissent distinctement. Sur ce graphe, supposons que l'on intervertisse, après l'application de la permutation B_N, les $N/2$ premiers LLRs avec les $N/2$ suivants (*i.e.* inverser les entrées des deux blocs en pointillés rouge), il peut être observé que cela revient à inverser les deux entrées de chaque bloc noyau du niveau $j = 0$, et ne modifie donc pas les LLRs genie-aided. Plus généralement, le même raisonnement peut s'appliquer à l'intérieur de chacun des deux blocs marqués en rouge, où l'on peut intervertir les $N/4$ premiers LLRs d'entrée avec les $N/4$ suivants sans modification de la sortie. Par récurrence, on peut associer à chaque niveau j exactement 2^j permutations des LLRs d'entrée telles que cela ne change pas les LLRs genie-aided obtenus.

Définition 3 *Pour tout* $j \in \{0,\dots,n-1\}$ *et* $k \in \{0,\dots,2^j-1\}$, *on appelle permutation élémentaire toute permutation* $\phi_{k,j} : \{0,\dots,N-1\} \to \{0,\dots,N-1\}$, *inversant les éléments des blocs* $[k,\dots,k+2^{n-1-j}-1]$

et $[k+2^{n-1-j},\ldots,k+2^{n-j}-1]$, *i.e.:*

$$\phi_{k,j}(i)=\begin{cases} i & si\ i<k\ ou\ i\geq k+2^{n-j} \\ (i+2^{n-1-j}) & si\ k\leq i<k+2^{n-1-j} \\ (i-2^{n-1-j}) & si\ k+2^{n-1-j}\leq i<k+2^{n-j} \end{cases} \tag{2.14}$$

où a mod b représente la valeur de a modulo b.

Ces permutations élémentaires s'appliquent à un code non-poinçonné, mais sont tout particulièrement utiles pour l'étude des motifs de poinçonnage et de raccourcissement, en permettant de regrouper deux motifs distincts dont les performances sont rigoureusement identiques.

Définition 4 *Deux motifs de poinçonnage P et P' sont dits équivalents, et notés $P\approx P'$, s'il existe une série de permutations élémentaires $\phi_{k_1,j_1},\ldots,\phi_{k_t,j_t}$ transformant $B_N(P)$ en $B_N(P')$:*

$$P'=B_N(\phi_{k_t,j_t}(\phi_{k_{t-1},j_{t-1}}(\ldots(\phi_{k_1,j_1}(B_N(P)))\ldots))) \tag{2.15}$$

La propriété suivante permet de valider l'égalité des performances de deux motifs équivalents:

Proposition 8 *Soit deux motifs équivalents P et P'. Soit $y[P]_0^{N-1}$ et soit $y'[P']_0^{N-1}$ la permutation de $y[P]_0^{N-1}$ par la permutation $B_N\circ\phi_{k_t,j_t}\circ\cdots\circ\phi_{k_1,j_1}\circ B_N$, où ϕ_{k_i,j_i} sont des permutations élémentaires telles que définies dans l'équation* (2.14)*. Pour tout $i\in\{0,\ldots,N-1\}$, soit $L[P]_i^{(ga)}$ et $L'[P']_i^{(ga)}$ les valeurs des LLRs obtenues à l'issue du décodage genie-aided de $y[P]_0^{N-1}$ et $y'[P']_0^{N-1}$ respectivement, en supposant le mot de code $u_0^{N-1}=\mathbf{0}_N$. Alors:*

$$L[P]_i^{(ga)}=L'[P']_i^{(ga)},\quad \forall i\in\{0,\ldots,N-1\} \tag{2.16}$$

Cette propriété est la conséquence immédiate des remarques précédentes. Deux motifs équivalents vérifient plusieurs conditions:

Proposition 9 *Si deux motifs de poinçonnage P et P' sont équivalents, alors:*

(i) Les distributions des LLRs genie-aided $\Gamma(W[P]_N^{(i)})$ et $\Gamma(W[P']_N^{(i)})$ d'un canal virtuel sont identiques pour tout canal W:

$$\Gamma(W[P]_N^{(i)})=\Gamma(W[P']_N^{(i)}),\quad \forall i\in\{0,\ldots,N-1\},\quad \forall W \tag{2.17}$$

(ii) Les coefficients de Bhattacharrya d'un canal virtuel sont identiques:

$$Z(W[P]_N^{(i)})=Z(W[P']_N^{(i)}),\quad \forall i\in\{0,\ldots,N-1\},\quad \forall W \tag{2.18}$$

(iii) Les motifs d'effacement sont identiques:

$$E[P]=E[P'] \tag{2.19}$$

Pour démontrer cette propriété, il suffit d'observer que (i) est la conséquence directe de la proposition précédente, que (ii) est celle de (i) d'après l'équation (1.38), et que (iii) est celle de (ii). En particulier, cette propriété prouve que toute méthode de construction du code par évolution de densité donnera systématiquement le même résultat pour les motifs P et P'.

TABLE 2.1: *Motifs non-équivalents ayant le même motif d'effacement*

motif de poinçonnage	$\mathcal{Z} = \{Z(W[P]_8^{(i)}) \mid i = 0,\ldots,7\}$ $p = \{p_{i\vert} \mid i = 0,\ldots,7\}$ $[W = \text{BEC}(\varepsilon = 0.5)]$
P	$\mathcal{Z} = \{\ 1,\ \ 1,\ \ 1,\ \ \mathbf{0.75},\ \ 1,\ \ \mathbf{0.625},\ \mathbf{0.5625},\ \mathbf{0.0625}\ \}$ $p = \{\ 0.5, 0.5, 0.5, \mathbf{0.375}, 0.5, \mathbf{0.3125}, \mathbf{0.2813},\ \mathbf{0.0313}\ \}$
P'	$\mathcal{Z}' = \{\ 1,\ \ 1,\ \ 1,\ \ \mathbf{0.75},\ \ 1,\ \ \mathbf{0.750},\ \mathbf{0.4376}, \mathbf{0.0.625}\ \}$ $p' = \{\ 0.5, 0.5, 0.5, \mathbf{0.375}, 0.5,\ \mathbf{0.375},\ \mathbf{0.2188},\ \mathbf{0.0313}\ \}$

$P = [11101000]$, $P' = [11011000]$, $E[P] = E[P'] = P$

Algorithme 3 Function ϕ

procédure $\phi(P)$
  pour $j = 0 : 1 : n-1$ **faire**
    pour $k = 0 : 2^{n-j} : N-1$ **faire**
      si $P_k^{k+2^{n-1-j}-1} <_{lex} P_{k+2^{n-1-j}}^{k+2^{n-j}-1}$ **alors**
        $P \leftarrow \phi_{k,j}(P)$ ▷ Permuter $P_k^{k+2^{n-1-j}-1}$ et $P_{k+2^{n-1-j}}^{k+2^{n-j}-1}$
      fin si
    fin pour
  fin pour
  return P
fin procédure

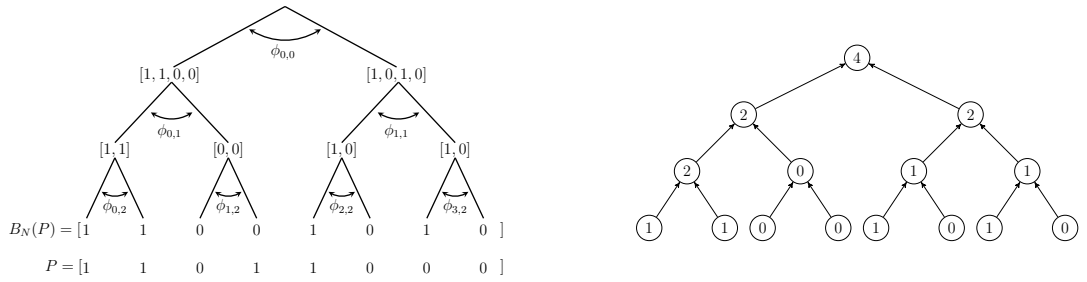
Mentionnons finalement qu'il est possible de trouver deux motifs P et P' tels que $E[P] = E[P']$, mais sans que ces motifs ne soient équivalents, contrairement à ce qui est indiqué dans certains articles. Un contre-exemple est fourni dans le tableau 2.1 pour un code polaire de longueur $N = 8$ sur un canal BEC avec $P = [1,1,1,0,1,0,0,0]$ et $P' = [1,1,0,1,1,0,0,0]$. Il apparaît que l'on a bien $E[P] = E[P'] = P$, mais que les valeurs des probabilités d'erreurs et des coefficients de Bhattacharrya sont différentes pour les indices $i = 5$ et $i = 6$.

Dans la suite, on appellera *classe d'équivalence* d'un motif de poinçonnage P l'ensemble des motifs $P' \approx P$. Notons qu'il existe un total de $N - 1 = \sum_{j=0}^{n-1} 2^j$ permutations élémentaires, soit potentiellement 2^{N-1} motifs équivalents; néanmoins un grand nombre de permutations possibles laissent le motif P invariant de sorte que la dimension d'une classe d'équivalence est nettement réduite (et dépend du motif P). Pour simplifier la recherche du meilleur motif, on se propose de définir un unique représentant pour chaque classe d'équivalence, appelé motif primitif dans la suite.

2.2.4 MOTIF PRIMITIF

Pour définir un représentant d'une classe d'équivalence, il convient de définir une règle relative aux permutations élémentaires à appliquer sur tout motif P de manière à ce que le motif obtenu soit identique pour tout membre d'une même classe d'équivalence. Une telle fonction est définie dans l'algorithme 3. Celui-ci décrit une procédure passant en revue toutes les permutations élémentaires $\phi_{k,j}$ dans un ordre particulier, mais n'applique une permutation élémentaire sur P que si $[P(k),\ldots,P(k+2^{n-1-j}-1)] <_{lex} [P(k+2^{n-1-j}),\ldots,P(k+2^{n-j}-1)]$, où $<_{lex}$ représente l'ordre lexicographique. A noter qu'appliquer la permutation $\phi_{k,j}$ revient en fait à permuter ces deux blocs. On définit alors le motif :

$$\Phi(P) = B_N(\phi(B_N(P))) \tag{2.20}$$



(a) Action des permutations sur $B_N(P)$ (b) Représentation de $B_N(P)$ sous forme d'arbre binaire

FIGURE 2.4: *Action des permutations et représentation en arbre du motif* $P = [1,1,0,1,1,0,0,0]$

Proposition 10 *Le motif* $\Phi(P)$ *vérifie les propriétés suivantes:*

(i) $\Phi(P) \approx P$

(ii) $P \approx P' \Leftrightarrow \Phi(P) = \Phi(P')$

Les démonstrations sont données en Annexe B.2. En particulier, celle-ci prouve que le motif $\Phi(P)$ ne dépend que de la classe d'équivalence de P. Dans la suite, on appellera *motif primitif* de la classe d'équivalence de P, le motif $P' \approx P$ tel que $\Phi(P') = P'$. Nécessairement, chaque classe d'équivalence contient un unique motif primitif.

L'action des permutations élémentaires sur un motif peut se visualiser par un arbre binaire. Un exemple pour le motif $P = [1,1,0,1,1,0,0,0]$ est présenté sur la figure 2.4(a). A partir de cette représentation, on peut utiliser le formalisme d'un arbre binaire pour représenter un motif, comme montré sur la figure 2.4(b). La racine de l'arbre est fixée à la valeur N_p, tandis que les valeurs des nœuds fils correspondent au nombre de positions poinçonnées dans les sous-motifs correspondants. Nécessairement, la somme des valeurs des fils est égale à la valeur du nœud parent. Notons que l'action de la fonction ϕ consiste à s'assurer que la valeur d'un fils de gauche est toujours supérieure ou égale à celle du fils de droite. En théorie des graphes, les arbres binaires associés à deux motifs équivalents sont appelés *arbres isomorphes*, tandis que l'arbre binaire associé à un motif primitif est souvent qualifié d'*arbre monotone*. Dès lors, générer l'ensemble des motifs primitifs pour un couple (N, N_p) donné revient à explorer l'ensemble des arbres binaires monotones, *i.e.* les arbres obtenus par toutes les décompositions de N_p telles que le fils de gauche soit supérieur ou égal à celui de droite.

La première conséquence de cette observation est que les motifs primitifs peuvent être obtenus de manière récursive:

Proposition 11 *Soit* $P_0, P_1 \in \{0,1\}^{\frac{N}{2}}$ *deux motifs de poinçonnage. Alors :*

$$P = B_N([B_{\frac{N}{2}}(P_0)\ B_{\frac{N}{2}}(P_1)]) \ \textit{ primitif } \Leftrightarrow P_0 \textit{ et } P_1 \textit{ primitifs et } B_{\frac{N}{2}}(P_0) \geq_{lex} B_{\frac{N}{2}}(P_1),$$

où $B_N(P)$ *représente la permutation à inversion de bit du vecteur binaire* P.

La deuxième conséquence est que le nombre total de motifs primitifs de longueur N, noté dans la suite $A(n)$, revient à dénombrer les arbres binaires monotones, dont le résultat est bien connu et donné par la suite récurrente:

$$\begin{cases} A(1) & = 3 \\ A(n+1) & = \dfrac{A(n) \cdot (A(n)+1)}{2} \end{cases} \tag{2.21}$$

dont les premières valeurs sont données dans le tableau 2.2. Le nombre de motifs de poinçonnage primitifs de longueur N et de poids $|P| = N_p$, noté $a(n, N_p)$, peut se calculer par récurrence d'après la formule suivante, démontrée dans [62, prop. 2] pour le cas du raccourcissement (on verra plus loin qu'il s'agit exactement du même problème de dénombrement):

$$\begin{cases} a(1, N_p) & = 1 \\ a(n, N_p) & = \left(\sum_{i=\max(0, N_p/2)}^{\lfloor N_p/2 \rfloor} a(n-1, i) \cdot a(n-1, N_p - i) \right) \\ & \quad - \frac{1}{2}\tau \cdot a(n-1, N_p/2) \cdot \left(a(n-1, N_p/2) - 1 \right) \end{cases} \tag{2.22}$$

avec $\tau = N_p \ (mod\ 2)$.

2.2.5 Motif symétrique

Nous proposons de nous intéresser à une sous-catégorie très particulière de motifs primitifs, appelés motifs symétriques, et pouvant être facilement caractérisés à partir de la matrice G_N.

Définition 5 *Un motif de poinçonnage est dit symétrique si $E[P] = P$*

Une caractérisation simple de ces motifs est donnée par la propriété suivante (pour la démonstration voir l'Annexe B.3):

Proposition 12 *Soit $P \subset \{0,1\}^N$ un motif de poinçonnage. La condition $E[P] = P$ est vérifiée si et seulement si $P = \mathbf{0}_N$ ou P est une réunion de lignes de la matrice G_N, i.e. $\exists \mathcal{L} \subset \{0, \dots, N-1\}$, $P = \bigvee_{i \in \mathcal{L}} G_N^{(i)}$.*

Les motifs symétriques sont des motifs primitifs:

Proposition 13 *Si P est un motif symétrique, alors $\Phi(P) = P$, i.e. P est le motif primitif de sa classe d'équivalence.*

La Proposition 7 indique que les motifs d'effacement sont nécessairement des réunions de lignes de G_N. La caractérisation des motifs symétriques ci-dessus prouve qu'il s'agit d'une condition nécessaire et suffisante, puisque, pour tout vecteur E réunion de lignes de G_N, le motif $P = E$ est tel que $E[P] = E$. Un corollaire est que, pour tout motif primitif non-symétrique P_{ns}, il existe toujours un motif symétrique P_s (nécessairement non équivalent avec P_{ns}) tel que $E[P_{ns}] = E[P_s]$.

Notons une propriété particulière des lignes de la matrice G_N:

Proposition 14 *La permutation à inversion de bit d'une ligne de G_N est une ligne de G_N:*

$$B_N(G_N^{(i)}) = G_N^{(j)}, \tag{2.23}$$

où j est tel que son développement binaire vérifie $b_n^{(j)}[k] = b_n^{(i)}[n-1-k]$, $\forall k \in \{0, \dots, n-1\}$.

Cette démonstration est fournie en Annexe B.3. La conséquence est que, si l'on appelle $\mathbb{S}$ l'ensemble des motifs symétriques, alors l'image de $\mathbb{S}$ par la permutation B_N est exactement $\mathbb{S}$. Cela permet de simplifier la génération des motifs symétriques de manière récursive :

Proposition 15 *Soit $P_0, P_1 \in \{0,1\}^{\frac{N}{2}}$ deux motifs de poinçonnage. Alors :*

$$P = [P_0\ P_1] \text{ symétrique } \Leftrightarrow P_0 \text{ et } P_1 \text{ symétriques et } P_1 \subseteq P_0$$

TABLE 2.2: *Nombre total de motifs primitifs et symétriques*

n	1	2	3	4	5	6	Série
Total motifs primitifs	3	6	21	231	26796	$> 3.59 \cdot 10^8$	$A(n+1) = \frac{A(n)A(n+1)}{2}$
Total motifs symétriques	3	6	20	168	7581	$> 7.8 \cdot 10^6$	Nombres de Dedekind

TABLE 2.3: *Dénombrement des motifs en fonction de N_p*

n	Motif	Dénombrement pour $N_p \in \{0,\dots,2^n\}$
1	primitifs	$\{1,1,1\}$
	symétriques	$\{1,1,1\}$
2	primitifs	$\{1,1,2,1,1\}$
	symétriques	$\{1,1,2,1,1\}$
3	primitifs	$\{1,1,3,3,5,3,3,1,1\}$
	symétriques	$\{1,1,3,3,4,3,3,1,1\}$
4	primitifs	$\{1,1,4,6,14,17,27,28,35,28,\dots\}$
	symétriques	$\{1,1,4,6,10,13,18,19,24,16,\dots\}$

A titre d'exemple, le motif $P = [1,1,0,1,1,0,0,0]$, présenté dans la figure 2.4 est non-symétrique, du fait que $[1,1,0,0] \not\preceq [1,0,1,0]$, mais reste primitif puisque $[1,1,0,0] \succeq [1,0,1,0]$.

Tout comme pour les motifs primitifs, on peut s'intéresser au dénombrement des motifs symétriques. Il s'avère que ce problème est lié à un problème de dénombrement bien connu et existant sous différentes formes, et dont la solution est donnée par les nombres de Dedekind. Le rapprochement peut être fait à partir de l'arbre présenté dans la figure 2.4 en considérant les ensembles des indices tels que $P(i) = 1$, formant un ensemble partiellement ordonné[1]. S'il existe bel et bien une formule analytique, celle-ci n'a qu'un intérêt limité du fait de la croissance extrêmement rapide de cette suite. Les premières valeurs sont données dans le tableau 2.2. Par contre, le nombre de motifs symétriques pour un N_p donné ne semble pas pouvoir s'obtenir simplement. Les résultats numériques présentés dans les tableaux 2.2 montrent que le nombre de motifs primitifs et symétriques croît très rapidement avec n de sorte qu'il n'est déjà plus envisageable d'explorer tous les motifs primitifs pour $n \leq 6$, tandis que les motifs symétriques peuvent à la rigueur être passés en revue jusqu'à $n = 6$.

2.2.6 MOTIFS DE RACCOURCISSEMENT

La classification introduite pour les motifs de poinçonnage peut également s'appliquer aux motifs de raccourcissement. Tout d'abord rappelons que les positions raccourcies sont des bits codés dont la valeur est connue, *i.e.* indépendante de tout bit d'information. Cela a pour conséquence que certains canaux virtuels ont une capacité symétrique égale à 1, mais sans que ces canaux puissent être utilisés pour transmettre de l'information. Pour tout motif de raccourcissement S, le motif $E[S]$ désigne les positions

[1] D'ailleurs, il a été proposé dans [7] une méthode pour calculer les nombres de Dedekind revenant exactement à dénombrer les motifs symétriques.

de ces canaux inutilisables. Soit i tel $S(i) = 1$. Puisque:

$$x_i = \sum_{G_N^{(i)}[N-1-j]=1} u_j, \tag{2.24}$$

il s'ensuit que l'ensemble des bits u_j tels que $G_N^{(i)}[N-1-j] = 1$ ne peuvent pas être des bits d'information. Autrement dit, $\pi_{lr}(G_N^{(i)}) \subset E[S]$, où $\pi_{lr} : \{0,\ldots,N-1\} \to \{N-1,\ldots,0\}$ est la permutation inversant les positions d'indices i et $N-1-i$. En appliquant ce même raisonnement sur tous les bits i tels que $S(i) = 1$, on obtient finalement que $\pi_{lr}(E[S])$ est nécessairement une réunion de lignes de G_N.

Pour mettre en évidence le parallèle entre le raccourcissement et le poinçonnage, il convient d'observer attentivement les règles de propagation des valeurs connues dans le cas d'un code raccourci:

$$I(W^-) = 1 \Leftrightarrow I(W_0) = 1 \text{ et } I(W_1) = 1$$
$$I(W^+) = 1 \Leftrightarrow I(W_0) = 1 \text{ ou } I(W_1) = 1,$$

et celles relatives aux effacements pour un code poinçonné:

$$I(W^-) = 0 \Leftrightarrow I(W_0) = 0 \text{ ou } I(W_1) = 0$$
$$I(W^+) = 0 \Leftrightarrow I(W_0) = 0 \text{ et } I(W_1) = 0,$$

Si l'on considère le motif de poinçonnage P et le motif de raccourcissement $S = \pi_{lr}(P)$, alors on a nécessairement $I(W[S]_N^{(i)}) = 1 \Leftrightarrow I(W[P]_N^{(\pi_{lr}(i))}) = 0$.

Toute la classification et les propriétés mises en évidence pour le poinçonnage se transposent au raccourcissement:

Définition 6 *On dira qu'une motif de raccourcissement S est:*

- *primitif si $\phi(S) = \pi_{lr}(S)$*
- *symétrique si $E[S] = S$*

Les propriétés suivantes sont immédiates:

Proposition 16 *Soit P un motif de poinçonnage et $S = \pi_{lr}(P)$ un motif de raccourcissement. Alors:*

- *$P \approx P' \Leftrightarrow S \approx S'$*
- *P primitif $\Leftrightarrow$ S primitif*
- *P symétrique $\Leftrightarrow$ S symétrique*

Une précision importante s'impose toutefois concernant le raccourcissement d'un code, et qui n'avait pas lieu d'être pour le poinçonnage. Si l'on considère que tous les bits de l'ensemble $E[S]$ ont des valeurs fixées à 0 ou 1 (*i.e.* exactement comme les bits figés), alors seuls les motifs symétriques sont à considérer. Il suffit pour s'en rendre compte de repartir de l'équation (2.24). Pour obtenir un motif de raccourcissement primitif non-symétrique, il est nécessaire que les valeurs des bits de $E[S]$ soient en fait des combinaisons particulières des bits d'information. Un exemple est donné dans la figure 2.5 pour le motif primitif non-symétrique $S = [0,0,0,1,1,0,1,1]$ tel que $E[S] = [0,0,0,1,0,1,1,1]$.

En conclusion, l'optimisation du motif de poinçonnage et du motif de raccourcissement est tout à fait similaire dans les deux cas et les mêmes méthodes peuvent être utilisées. Précisons toutefois que, malgré leurs similitudes, il n'est pas possible en général d'assimiler un code poinçonné à un code raccourci, de sorte que trouver le meilleur code pour un couple (N,K) donné avec $N \neq 2^n$ nécessite d'explorer les deux techniques parallèlement. Cette dernière affirmation sera justifiée plus loin.

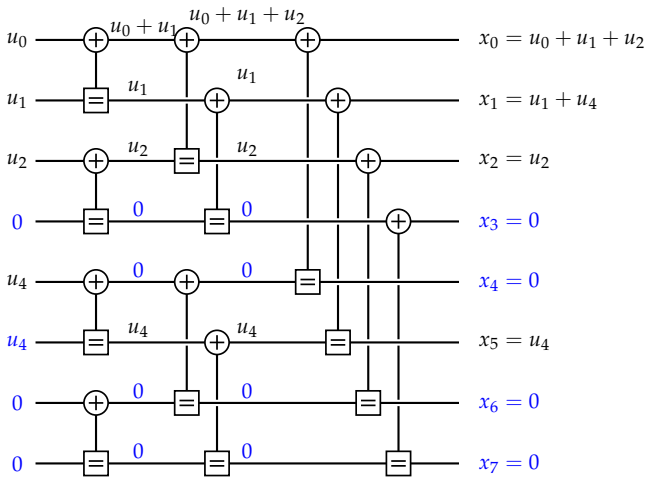


FIGURE 2.5: *Raccourcissement d'un code par un motif primitif non-symétrique : le motif de raccourcissement* $S = [0,0,0,1,1,0,1,1]$ *ne peut être obtenu que lorsque* $u_5 = u_4$.

2.2.7 MOTIFS DE L'ÉTAT DE L'ART

On se propose de revenir sur les principaux motifs de l'état de l'art à la lueur de la classification introduite ci-dessus.

Tout d'abord, le motif de poinçonnage QUP (poinçonnage des N_p premières positions) est clairement un motif symétrique, puisqu'il s'agit de la réunion des N_p premières lignes de G_N (d'après la proposition 13). Le motif de raccourcissement QUS (raccourcissement des N_s dernières positions) est obtenu via la permutation π_{lr} du motif de poinçonnage QUP, il est donc également symétrique. Une autre solution proposée est d'utiliser le motif obtenu par la permutation à inversion du motif QUP [15]. Ce motif est symétrique, puisqu'il est également une réunion de lignes de G_N. Ce résultat s'étend également au motif de raccourcissement obtenu par la permutation à inversion de bit du motif de raccourcissement QUS.

Considérons maintenant les motifs obtenus en poinçonnant les canaux virtuels les moins fiables du code mère [109], ou en raccourcissant les plus fiables [82]. En Annexe B.3, nous démontrons que ces deux motifs sont également symétriques. La démonstration exploite le fait que les canaux virtuels du code mère respectent l'ordre partiel décrit par l'équation (1.15).

Finalement, précisons que la solution appliquée dans la 5G n'est pas associable à l'un des motifs précédents, du fait que l'entrelaceur par bloc appliqué après l'encodage, mais avant de pratiquer le poinçonnage ou le raccourcissement. Cependant, une analyse méticuleuse de cet entrelaceur permet de s'assurer que cela revient à appliquer un motif de poinçonnage ou de raccourcissement symétrique directement après l'encodage. Cette propriété de symétrie est d'ailleurs exploitée lors du choix des positions des bits figés, afin de s'assurer que les canaux virtuels de capacité nulle (pour le poinçonnage) ou de capacité égale à 1 (pour le raccourcissement) ne sont pas utilisés pour transmettre des bits d'information.

2.3 ÉNUMÉRATION DES MOTIFS DE POINÇONNAGE ET DE RACCOURCISSEMENT

La section précédente a permis d'identifier une liste de motifs de poinçonnage, appelés motifs primitifs, tous non-équivalents, et significativement moins nombreux par rapport au nombre total de motifs de poinçonnage. Elle a également mis en évidence le fait que tout motif de poinçonnage primitif peut être transformé en un motif de raccourcissement primitif en appliquant la permutation π_{lr}. L'étape suivante

consiste à s'efforcer d'identifier, parmi tous ces motifs, celui donnant les meilleures performances pour le décodage SC d'un code polaire $\mathcal{C}(N,K)$ avec $N \neq 2^n$. Cependant, on a vu que le nombre de motifs primitifs augmente très rapidement avec n, de sorte qu'une exploration exhaustive n'est possible que pour des valeurs de N et/ou de N_p assez faibles. Pour les codes longs, nous proposons également une méthode se concentrant sur un sous-ensemble de motifs bien caractérisés et faciles à passer en revue grâce à la classification introduite précédemment.

2.3.1 Analyse exhaustive

Différentes méthodes sont possibles pour énumérer les motifs primitifs. La première consiste à les construire de manière récursive d'après la propriété 6. La limite de cette méthode est qu'elle requiert de stocker en mémoire tous les motifs primitifs de longueurs $N/2$ de poids $|P| \leq N_p$. Une autre approche ne nécessitant pas d'une grande quantité de mémoire et pouvant fonctionner également pour N élevé et N_p faible, est d'énumérer les $\binom{N}{N_p}$ motifs de poids $|P| = N_p$ et d'écarter tous les motifs non primitifs. Notons que ces deux approches peuvent également être utilisées pour énumérer les motifs symétriques.

Une fois un motif primitif obtenu, l'étape suivante consiste à estimer la performance du SC. Pour cela, les techniques d'évolution de densité présentées dans le chapitre précédent constituent un outil précieux. Elles sont en effet capables de fournir, pour un code poinçonné aussi bien que raccourci, une estimation précise des performances du SC, d'après la formule:

$$\mathrm{WER}(\mathcal{C}) \approx 1 - \prod_{i \in \mathcal{I}} (1 - p_{i|}), \tag{2.25}$$

où $p_{i|} = \Pr(\hat{u}_i \neq u_i \mid u_0^{i-1})$. Ces probabilités sont calculées en tenant compte du fait que le code est poinçonné ou raccourci. Dans la mesure où l'on projette de passer en revue un grand nombre de motifs, les méthodes par approximation gaussienne (pour un canal AWGN) sont celles ayant la plus faible complexité.

La figure 2.6 compare les performances des meilleurs motifs obtenus par poinçonnage et par raccourcissement pour un code de longueur $N = 64$ avec $N_p = 10$ positions poinçonnées/raccourcies sur un canal AWGN. Pour chaque motif primitif, une recherche par dichotomie est appliquée de manière à déterminer le SNR tel que le taux d'erreur du SC estimé par DE-GA soit égal à 10^{-4} (à 0.01dB près). La figure présente les meilleurs motifs de poinçonnage et de raccourcissement. Les résultats corroborent ceux présentés dans [15], et pris en compte dans la 5G, à savoir que le poinçonnage semble préférable pour des rendements inférieurs à $\frac{1}{2}$, tandis que le raccourcissement semble préférable dans le cas inverse.

2.3.2 Application aux codes longs

Dans [62], l'exploration des motifs de raccourcissement utilise un algorithme de Stack pour tenter d'orienter la recherche vers les meilleurs motifs. La métrique est basée sur une estimation du taux d'erreur des sous-motifs primitifs de poids $N/2^k$. Cependant, cet algorithme requiert une très grande quantité de mémoire pour être réellement efficace et des calculs supplémentaires sont nécessaires durant le processus d'exploration.

Nous proposons une autre approche, consistant à se concentrer sur un sous-ensemble de motifs primitifs. Tout d'abord, nous proposons de limiter la recherche aux motifs symétriques. Il sera montré plus loin que cela n'entraîne qu'une perte légère en terme d'optimalité. Cependant, le nombre de motifs symétriques reste encore impraticable pour des valeurs de N élevées et il convient de déterminer un critère pour réduire encore davantage la zone de recherche. Pour cela, on propose d'utiliser la caractérisation des motifs symétriques comme réunion de lignes de la matrice G_N et on définit l'*ordre* d'un motif symétrique, noté

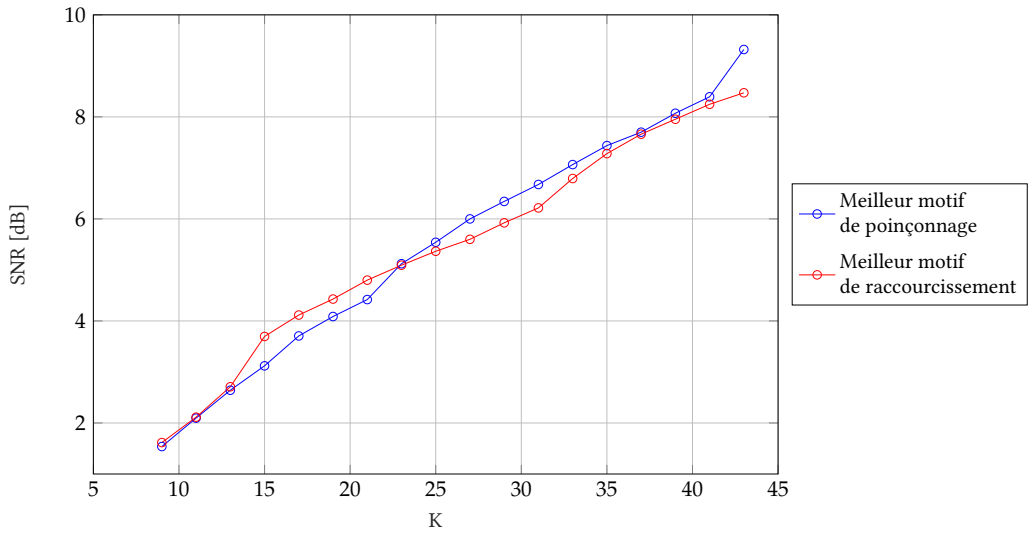


FIGURE 2.6: *Comparaison poinçonnage et raccourcissement pour* $N = 64$ *et* $N_p = 10$ *et un taux d'erreur cible de* 10^{-4}. *Pour chaque triplet* (N, K, N_p) *les motifs primitifs sont exhaustivement explorés et l'on détermine par DE-GA le SNR le plus faible garantissant un taux d'erreur de* 10^{-4}

λ, par:

$$\lambda = \min\{|\mathcal{L}| \mid P = \bigvee_{i \in \mathcal{L}} G_N^{(i)}\} \tag{2.26}$$

Nous proposons de rechercher tous les motifs symétriques d'ordre $\lambda \leq \lambda_{\max}$ où $\lambda_{\max}$ est une valeur choisie de manière à ce que le nombre de motifs considérés reste abordable. L'avantage d'une telle approche est que l'exploration peut s'implémenter simplement et ne requiert que très peu de mémoire. L'algorithme proposé utilise la caractérisation des motifs symétriques à partir des lignes de la matrices G_N. Plus précisément, un motif symétrique est défini par la liste des indices des lignes permettant de l'obtenir, notée $L = (L(1), L(2), \ldots, L(\lambda_{max}))$, avec $0 \leq L(\tau) \leq N$ pour tout $\tau \in \{1, \ldots, \lambda_{max}\}$. Pour tout $\lambda \in \{1, \ldots, \lambda_{max}\}$, on note $P_\lambda = \bigvee_{\tau \in \{1,\ldots,\lambda\}} G_N^{(L(\tau))}$. Ainsi, $L(1), \ldots, L(\lambda)$ sont les indices des λ lignes de G_N permettant d'obtenir le motif symétrique P_λ. Puisque les lignes de G_N sont indexées de 0 à $N-1$, on notera $G_N^{(N)} = \mathbf{0}_N$ le vecteur constitué de N zéros, et l'on définit également $P_0 = \mathbf{0}_N$. Pour une valeur de N_p et de λ_{max} donnée, l'algorithme proposé fonctionne de la manière suivante:

0. Initialiser $\lambda = 1$ et la liste $\mathcal{L}$ par $\mathcal{L}(\tau) = N$, pour tout $\tau \in \{1, \ldots, \lambda_{max}\}$ (équivalent à $P_\lambda = \mathbf{0}_N$)
1. Calculer les poids complémentaires pour chaque ligne de G_N, définis par:

$$W_{cp}^{(i)} = |G_N^{(i)}| - |G_N^{(i)} \vee P_{\lambda-1}|, \forall i \in \{0, \ldots, N-1\} \tag{2.27}$$

2. Déterminer le plus grand indice i strictement inférieur à $L(\lambda)$ tel que:

$$\begin{cases} 0 < W_{cp}(i) \leq N_p - |P_\lambda|, & \text{si } \lambda < \lambda_{max} \\ W_{cp}(i) = N_p - |P_\lambda|, & \text{si } \lambda = \lambda_{max} \end{cases} \tag{2.28}$$

Si aucun indice i ne vérifie l'équation (2.28), alors $\lambda = \lambda - 1$ (remonter au niveau supérieur) et aller directement à l'étape 4).

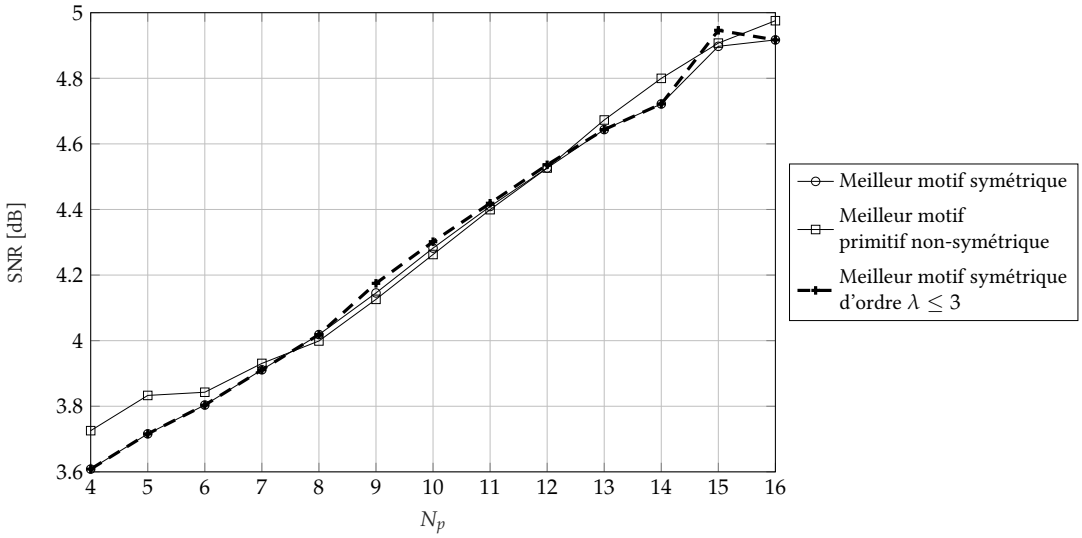


FIGURE 2.7: *Comparaison des motifs de poinçonnage primitifs et symétriques pour $N = 64$ et $K = 20$ et un taux d'erreur cible de 10^{-4}. Pour chaque triplet (N, K, N_p) les motifs primitifs sont exhaustivement explorés et l'on détermine par DE-GA le SNR le plus faible garantissant un taux d'erreur de 10^{-4}*

3. Mettre à jour la valeur $\mathcal{L}(\lambda) = i$.
   - Si $|P_\lambda| = |N_p|$, alors conserver P_λ (motif primitif symétrique d'ordre λ et de poids N_p), et retourner à l'étape 2) pour continuer la recherche sur le même niveau;
   - Sinon, incrémenter $\lambda = \lambda + 1$, et revenir à l'étape 1) (continuer la recherche sur le niveau inférieur).
4. Si $\lambda = 0$ alors tous les motifs ont été trouvés et le processus s'arrête; autrement, retourner à l'étape 1).

Il peut être observé que la complexité de cet algorithme est de l'ordre de $\mathcal{O}(N^{\lambda_{\max}})$ de sorte que celle-ci reste abordable même pour des codes longs lorsque la valeur de $\lambda_{\max}$ reste limitée.

Proposition 17 *Le motif QUP est un motif symétrique d'ordre $\lambda_{QUP} = \sum_{k=0}^{n-1} b_n^{(N_p)}[k] \leq n$*

La démonstration est donnée en Annexe B.4. La conséquence est que si l'on choisit $\lambda_{max} \geq \lambda_{QUP}$, alors l'algorithme garantit une performance au moins aussi bonne que celle du motif QUP.

La figure 2.7 compare les performances des motifs primitifs symétriques et non-symétriques pour un code de longueur $N = 64$ et $K = 20$ en fonction de N_p pour un canal AWGN. Pour chaque triplet (N, K, N_p) et chaque motif de poinçonnage, on détermine par dichotomie le SNR tel que l'estimation du taux d'erreur du SC par DE-GA soit égale à 10^{-4}. Les résultats présentés montrent que le fait de restreindre l'exploration des motifs aux motifs symétriques n'induit au pire qu'une perte minime d'optimalité. De plus, il apparaît également qu'introduire une limitation sur l'ordre (en l'occurrence $\lambda \leq \lambda_{max} = 3$)ne pénalise pas significativement les performances.

Les analyses ci-dessus sont uniquement basées sur des estimations par DE-GA. La figure 2.8 fournit une validation de la méthode proposée pour des codes de longueurs $N = 256$ et $N = 1024$. Pour $N = 256$ et $N_p = 85$, la méthode proposée fait très légèrement mieux que le poinçonnage QUP pour $\lambda = 5 > \lambda_{QUP} = 4$, tandis que pour $N = 1024$ et $N_p = 336$, celle-ci fait aussi bien que le poinçonnage QUP pour $\lambda = \lambda_{QUP} = 3$.

Malgré tout, pour les codes longs ou des valeurs de N_p élevées, cet algorithme risque de nécessiter un long processus d'optimisation. Il faut donc se tourner vers les approches alternatives.

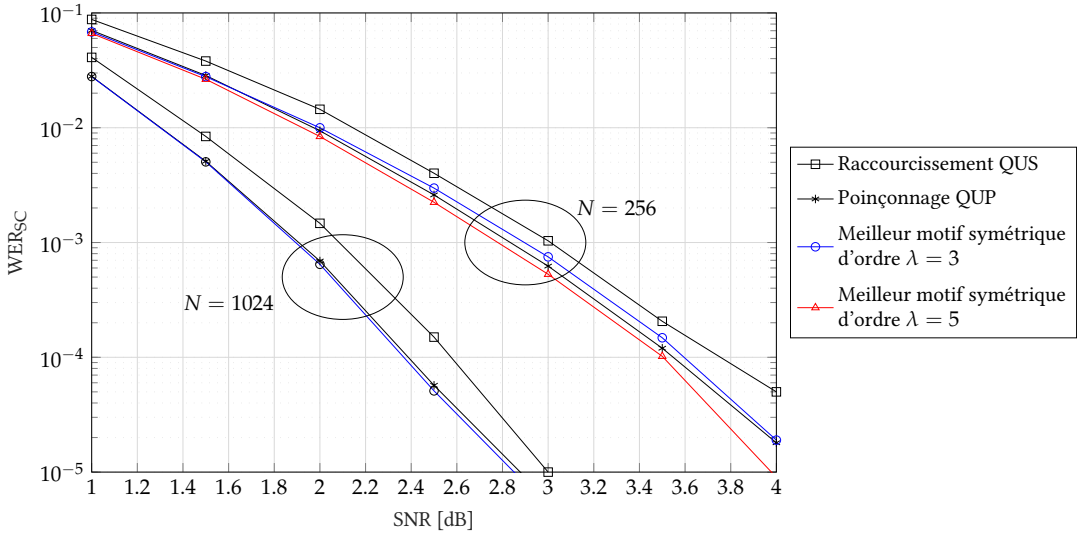


FIGURE 2.8: *Comparaison des motifs de poinçonnage pour* $(N = 1024, K = 256, N_p = 336)$ *et* $(N = 256, K = 64, N_p = 85)$ *sur un canal AWGN.*

2.4 POINÇONNAGE, RACCOURCISSEMENT ET STRUCTURES MULTI-NOYAUX

2.4.1 MATRICE DE TRANSFORMATION D'UN CODE POINÇONNÉ OU RACCOURCI

Tout comme la matrice G_N permet de décrire l'opération d'encodage d'un code polaire par $x_0^{N-1} = u_0^{N-1} \cdot G_N$, il est possible de définir la *matrice de transformation* d'un code poinçonné, notée $G[P]_N$, décrivant l'opération d'encodage du code $x[P]_0^{N-1-N_p} = u[E]_0^{N-1-N_p} \cdot G[P]_N$, où $u[E]_0^{N-1-N_p}$ correspond à la séquence des u_0^{N-1} privée des bits effacés, et $x[P]_0^{N-1-N_p}$ correspond à la séquence des bits codés privée des bits poinçonnés. Pour tout motif de poinçonnage P, cette matrice s'obtient en supprimant les colonnes de la matrice G_N dont les indices sont tels que $P(i) = 1$ et les lignes dont les indices sont tels que $E[P](i) = 1$ [91]. Cependant, il est clair qu'il est possible de se restreindre aux motifs primitifs, dans la mesure où les matrices de deux motifs équivalents peuvent s'obtenir l'une à partir de l'autre par les opérations de permutations de colonnes ou de remplacement d'une ligne d'indice i par la somme de celle-ci avec une ligne d'indices $j > i$. On a vu notamment dans la Section 2.1.3 que ces opérations ne modifient pas l'exposant d'erreur ni l'exposant d'échelle d'une matrice.

Pour ce qui est du raccourcissement, il convient de mettre à part le cas des motifs primitifs non-symétriques (et leurs motifs équivalents), du fait que les bits raccourcis sont des combinaisons des bits d'information, de sorte que la matrice de transformation ne peut pas être obtenue immédiatement. Par contre, pour un motif de raccourcissement symétrique (ou équivalent) S, la matrice de transformation s'obtient en supprimant les colonnes de la matrice G_N dont les indices sont tels que $S(i) = 1$ et les lignes dont les indices sont tels que $E[S](i) = 1$. Nécessairement, dans le cas d'un motif symétrique, la matrice de transformation est triangulaire inférieure.

Dans cette partie, on propose d'étudier l'optimisation du motif de poinçonnage ou de raccourcissement d'après le critère de l'exposant d'erreur des matrices de transformation. Pour cela, on considérera la matrice de transformation d'un code poinçonné ou raccourci comme une matrice noyaux F_ℓ (au sens de la Section 2.1.3) de taille $\ell \times \ell$, où $\ell = N - N_p$. Dans la mesure où ces "noyaux" peuvent être décodés avec une faible complexité, il est attendu que l'exposant d'erreur obtenu ne pourra pas atteindre celui des meilleures matrices connues, mais servira de point de référence basse pour l'approche plus générale des

structures multi-noyaux.

2.4.2 Combinaison de plusieurs noyaux

Proposition 18 *Soit deux matrices de transformations $F_1 = G[P_1]_{N_1}$ et $F_2 = G[P_2]_{N_2}$ obtenues à partir de motifs de poinçonnage symétriques P_1 et P_2 de longueurs N_1 et N_2. Alors, la structure multi-noyaux $F_1 \otimes F_2$ est équivalente à un code de longueur $N_1 \cdot N_2$ poinçonné par le motif symétrique:*

$$P = (P_1 \otimes \mathbf{1}_{N_2}) \vee (\mathbf{1}_{N_1} \otimes P_2) = P_1 \otimes \mathbf{1}_{N_2} + \mathbf{1}_{N_1} \otimes P_2 - P_1 \otimes P_2 \tag{2.29}$$

Le même résultat s'applique pour deux motifs de raccourcissement symétriques S_1 et S_2 avec:

$$S = (S_1 \otimes \mathbf{1}_{N_2}) \vee (\mathbf{1}_{N_1} \otimes S_2) = S_1 \otimes \mathbf{1}_{N_2} + \mathbf{1}_{N_1} \otimes S_2 - S_1 \otimes S_2 \tag{2.30}$$

La démonstration est donnée en Annexe B.5.2. Cette proposition peut se réutiliser récursivement pour obtenir le motif de poinçonnage (ou de raccourcissement) pour une structure multi-noyaux $G[P_1]_{N_1} \otimes G[P_2]_{N_2} \otimes \cdots \otimes G[P_m]_{N_m}$ avec un nombre quelconque de noyaux distincts. En revanche, cette proposition ne s'applique pas dans le cas d'un motif primitif non-symétrique.

Cette propriété peut s'utiliser dans les deux sens. Tout d'abord, elle permet d'interpréter un code poinçonné ou raccourci comme une structure multi-noyaux, et permettant ainsi d'utiliser l'exposant d'erreur des matrices de transformation comme critère pour discriminer deux motifs. Inversement, elle permet, sous certaines conditions, de considérer une structure multi-noyaux comme un code poinçonné ou raccourci de sorte que la question du décodage efficace du code en devient considérablement simplifiée. Précisons cependant que la longueur du code poinçonné équivalent à une structure multi-noyaux peut être nettement plus longue que celle du code multi-noyaux. Par exemple, pour le noyau $F_3 = G[P]_4$ de taille $\ell = 3$ avec $P = [1,0,0,0]$, le code poinçonné équivalent à $F_3 \otimes F_3 \otimes F_3$ est de longueur $N = 64$ avec $N_p = 37$ positions poinçonnées.

Notons deux applications particulières de la proposition précédente:

Proposition 19

- *Appliquer le produit de Kronecker par G_2 à gauche (i.e. $G[P]_{N_1} = G_2$) est équivalent à répéter le motif P_2 puisque cela revient à prendre $P_1 = [0,0]$:*

$$P = \mathbf{1}_2 \otimes P_2 = [P_2\ P_2] \tag{2.31}$$

  Plus généralement, répéter cycliquement 2^k fois le motif P_2 revient à multiplier à gauche par $G_2^{\otimes k}$.
- *Appliquer le produit de Kronecker par G_2 à droite (i.e., $G[P_2]_{N_2} = G_2$) revient à choisir $P_2 = [0,0]$:*

$$P = P_1 \otimes \mathbf{1}_2 \tag{2.32}$$

Cette dernière proposition peut être appliquée au motif QUP. Supposons un code de longueur $N = 2^n$, poinçonné de $N_p = 2^k \cdot N'_p$ positions par le motif QUP, alors:

$$\begin{aligned} P_{QUP} &= [\underbrace{1,\ldots,1}_{2^k \cdot N'_p \text{ fois}}, 0,\ldots,0\,] \\ &= [\underbrace{1,\ldots,1}_{N'_p \text{ fois}},0,\ldots,0\,] \otimes [\underbrace{1,\ldots,1}_{2^k \text{ fois}}] \end{aligned}$$

Ainsi, la structure du code poinçonné par le motif QUP peut s'écrire $F_{N'_p} \otimes G_2^{\otimes k}$ où $F_{N'_p}$ est la matrice de transformation du code de longueur $N' = 2^{n-k}$ poinçonné de N'_p positions par le motif QUP. Ce résultat peut être utilisé pour simplifier la détermination de l'exposant d'erreur d'une matrice obtenue par le poinçonnage QUP en utilisant l'équation (2.13).

Un autre motif utilisé dans l'état de l'art est le motif obtenu par la permutation à inversion de bit du motif QUP, noté RQUP. Cette permutation peut s'écrire récursivement par $B_N = [2B_{N/2}\ B_{N/2} + 1]$. La propriété qui nous intéresse ici est de noter qu'une telle permutation regroupe les valeurs d'indices pairs sur la première moitié et ceux d'indices impairs sur l'autre moitié. Dès lors, si l'on suppose $N_p = 2N'_p$, alors on a:

$$\begin{aligned} P_{RQUP}(N, N_p) &= B_N(P_{QUP}(N, N_p)) \\ &= [B_{N/2}(P_{QUP}(N/2, N_p/2))\ B_{N/2}(P_{QUP}(N/2, N_p/2))] \\ &= [1,1] \otimes P_{RQUP}(N/2, N_p/2) \end{aligned}$$

Il s'ensuit que le motif RQUP est, pour sa part, factorisable par G_2 à gauche. Par récurrence, il est possible de généraliser à $N_p = 2^k \cdot N'_p$. Mentionnons que les mêmes observations peuvent être obtenues pour le cas du raccourcissement d'un code par le motif QUS et par le motif correspondant à la permutation à inversion de bit du motif QUS.

2.4.3 Exposant d'erreur et distance minimale d'un code raccourci

Ainsi que présenté dans le début de cette section, la matrice de transformation d'un code poinçonné ou raccourci peut être vue comme une matrice noyau d'un code polaire. Dès lors, on peut s'interroger sur la valeur de son exposant d'erreur. La proposition suivante permet d'obtenir les valeurs des distances partielles pour le cas d'un code raccourci par un motif symétrique:

Proposition 20 *Soit un code polaire raccourci par un motif symétrique S de longueur N et poids $|S| = N_s$. Alors les distances partielles sont données par:*

$$D[S]_i = |G[S]_N^{(i)}| = |G_N^{(i)}|, \ \forall\, i \in \{0, \ldots, N - N_s\} \tag{2.33}$$

La démonstration est donnée en Annexe B.5.3. L'exposant d'erreur s'obtient finalement d'après l'équation (2.11). Dans le cas particulier du motif de raccourcissement QUS, celui-ci s'écrit simplement:

$$\begin{aligned} \mathbf{E}(G[S]_N) &= \frac{1}{\ell} \sum_{i=0}^{\ell-1} \log_\ell(2^{|B_n^{(i)}|}) \\ &= \frac{1}{\ell \cdot \log_2(\ell)} \sum_{i=0}^{\ell-1} |B_n^{(i)}| \end{aligned}$$

A noter que l'on a $\mathbf{E}(G[S]_N) \leq \frac{1}{2}$ avec égalité si et seulement si $N - |S|$ est une puissance de deux (pour la démonstration voir les travaux sur la somme des développements binaires, par exemple dans [56, Th. 9]).

La distance minimale d'un code s'obtient d'après le minimum des distances partielles sur les indices $i \in \mathcal{I}$ [55]. Ce résultat, associé à la proposition précédente, permet à la fois de calculer simplement la distance minimale d'un code raccourci, et de définir le critère de choix de $\mathcal{I}$ de manière à garantir la meilleure distance minimale possible, à savoir sélectionner les lignes de poids les plus élevés.

Dans [16], une méthode est proposée afin d'optimiser la distance minimale d'un code polaire basé sur une structure multi-noyaux, consistant en fait à maximiser le minimum des distances partielles du

code, en exploitant le fait que les distances partielles du produit $F_1 \otimes F_2$ s'obtiennent simplement à partir des distances partielles des matrices noyaux F_1 et F_2. La Proposition 20 relève du même principe, mais appliqué à un code raccourci. Mentionnons qe l'un des exemples de [16] utilise une structure s'écrivant comme $G_2^{\otimes n_1} \otimes F_3^{\otimes n_2} \otimes G_2^{\otimes n_3}$, où $F_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. Or en utilisant les opérations n'affectant pas l'exposant d'erreur, décrites dans la section 2.1.3, il s'avère que cette matrice se réécrit comme $G[S_0]_4$ avec $S_0 = [0,0,0,1]$. Puisque ce motif est symétrique, et en utilisant la Proposition 19, la structure multi-noyaux est équivalente à un code raccourci par le motif $S = [1,1]^{\otimes n_1} \otimes S_0^{\otimes n_2} \otimes [1,1]^{\otimes n_3}$, de sorte que l'on peut utiliser la Proposition 20 de manière tout à fait équivalente.

Finalement, dans [74], une métrique est proposée pour optimiser le motif de raccourcissement, revenant à choisir celui-ci de manière à maximiser la moyenne géométrique des poids des lignes de la matrice $G[S]_N$, soit autrement dit, d'après la proposition 20, maximiser l'exposant d'erreur de la matrice $G[P]_N$. En particulier, il est démontré que le motif QUS constitue l'une des solutions offrant l'exposant d'erreur maximal [74, Th. 11]. La même approche est également employée pour le poinçonnage, où il est montré que le motif QUP est également optimal du point de vue de la métrique considérée. Cependant, pour garantir que ce résultat se retrouve au niveau de l'exposant d'erreur, il est nécessaire de prouver que les distances partielles du code poinçonné s'obtiennent également d'après les poids des lignes de la matrice de transformation $G[P]_N$. Si les simulations corroborent cette hypothèse, ce résultat reste encore à prouver.

2.5 Conclusion et perspectives

Ce chapitre s'est attelé à établir un panorama des méthodes pour obtenir une flexibilité sur la longueur du code. Il est tout d'abord établi que les techniques de poinçonnage et le raccourcissement, exclusivement abordées de manière séparées dans l'état de l'art, sont en réalité connexes et peuvent être traitées conjointement. Il est de plus mis en évidence que, malgré ce rapprochement, les solutions de l'une et l'autre méthode sont, sauf cas triviaux, distinctes et que les deux sont à considérer. A partir des permutations élémentaires du graphes polaires, les motifs équivalents sont caractérisés et un unique représentant de chaque classe d'équivalence est défini, appelé motif primitif. Une sous-catégorie de motifs primitifs, appelée motifs symétriques, est introduite et caractérisée précisément à partir des lignes de la matrice G_N. Les problèmes de dénombrement et d'énumération de ces deux catégories de motifs sont abordées, et un algorithme est proposé afin de trouver le meilleur motif – de poinçonnage ou de raccourcissement – pour les codes courts. Pour les codes longs, un algorithme est proposé, se concentrant sur un sous-ensemble de motifs symétriques, pour une perte limitée en terme d'optimalité.

Dans un second temps, le parallèle entre un code poinçonné ou raccourci et une structure multi-noyaux est explicité via la matrice de transformation du code. Il est montré qu'une structure multi-noyaux utilisant des noyaux obtenus par poinçonnage ou raccourcissement symétrique est équivalente à un code poinçonné ou raccourci respectivement. Une propriété est ensuite fournie indiquant que les distances partielles d'un code raccourci par un motif symétrique s'obtiennent d'après les lignes de la matrice de transformation. Ce résultat permet notamment de déterminer le critère pour optimiser la distance minimale du code raccourci, à savoir utiliser les lignes de poids maximal. Le lien avec les résultats de [74] est également souligné et permet d'affirmer que la solution QUS est celle offrant l'exposant d'erreur maximal parmi tous les motifs de raccourcissement symétriques. Toutefois, la transposition de ces résultats au poinçonnage reste encore un problème ouvert.

Notons également que l'exposant d'erreur est un critère essentiellement asymptotique, dont la valeur en longueur finie doit être relativisée. En effet, il n'est pas garanti qu'un code ayant un exposant d'erreur supérieur donne nécessairement de meilleures performances pour le décodeur SC, de sorte que la question

du critère pour choisir les matrices noyaux reste un problème ouvert. Si l'exposant d'échelle a notamment été discuté dans [34], il manque encore une comparaison précise entre les performances des noyaux optimaux du point de vue de l'exposant d'erreur et d'échelle.

3 Décodage à inversion

Contenu du Chapitre

Les deux principaux algorithmes offrant des performances significativement supérieures au décodeur par annulation successive (SC pour successive cancellation, en anglais) en longueur finie sont les décodeurs par liste, avec ou sans CRC (SCL pour successive cancellation list, en anglais, ou CA-SCL pour CRC-Aided SCL), et le décodeur SCS (pour Successive Cancellation Stack, en anglais). Une troisième est dénommée le décodage à inversion pour les codes polaires, et permet d'atteindre d'excellentes performances tout en offrant une complexité réduite. En opérant des tentatives successives, ce décodeur se contente en effet d'une unique tentative identique à celle du SC, lorsque celle-ci est correcte. Par contre, en cas d'erreur lors de cette tentative initiale, le choix des suivantes devient crucial pour garantir de bonnes performances et une complexité moyenne faible.

3.1 Principe du décodage à inversion

Le décodage à inversion [3] est une alternative aux décodeurs CA-SCL et SCS (cf section 1.4), et permet d'améliorer significativement les performances du SC, tout en conservant une complexité moyenne proche de celle du SC. Tout comme les deux autres décodeurs, le décodage à inversion requiert un système concaténé CRC-polaire et exploite le CRC pour déterminer le succès ou l'échec d'une tentative. Le décodage à inversion commence par effectuer le décodage SC, puis, en cas d'échec, effectue des tentatives de décodage successives, tant que le le CRC n'est pas vérifié, ou qu'un nombre maximal de T tentatives additionnelles n'a pas été atteint. Les nouvelles tentatives sont obtenues en inversant une ou plusieurs décisions par rapport au chemin du SC. Une notion très utile pour formaliser le décodage à inversion est de décrire toute trajectoire par rapport à celle du SC, au moyen d'une *inversion binaire* (d'ordre ω):

Définition 7 *Une inversion binaire d'ordre $\omega \in \{0,\dots,K\}$ est une série de ω indices $\mathcal{E} = \{i_1,\dots,i_\omega\} \subset \mathcal{I}$ tels que $i_1 < i_2 < \cdots < i_\omega$. Notons que pour $\omega = 0$, on a par définition $\mathcal{E} = \emptyset$.*

A toute inversion binaire $\mathcal{E}$ est associée une tentative de décodage, notée $\mathrm{SC}(\mathcal{E})$, similaire au décodage SC présenté dans l'algorithme 1, mais en modifiant le critère de prise de décision dure tel que donné dans l'équation (1.31) par le suivant:

$$\hat{u}[\mathcal{E}]_i = h_{\mathcal{I}}(L_i[\mathcal{E}]) \stackrel{\text{def}}{=} \begin{cases} u_i & \text{if } i \notin \mathcal{I} \\ \frac{1-\mathrm{sign}(L_i[\mathcal{E}])}{2} & \text{if } i \in \mathcal{I} \setminus \mathcal{E} \\ \frac{1+\mathrm{sign}(L_i[\mathcal{E}])}{2} & \text{if } i \in \mathcal{E}, \end{cases} \tag{3.1}$$

où

$$L_i[\mathcal{E}] = \log\left(\frac{\Pr(u_i = 0 \mid y_0^{N-1}, \hat{u}[\mathcal{E}]_0^{i-1})}{\Pr(u_i = 1 | y_0^{N-1}, \hat{u}[\mathcal{E}]_0^{i-1})}\right), \tag{3.2}$$

sont les LLRs calculés en utilisant les décisions dures $\hat{u}[\mathcal{E}]_0^{i-1}$. Notons que ces LLRs sont identiques à ceux calculés par le décodage SC jusqu'à la toute première position $i_1 \in \mathcal{E}$, mais ils différent pour $i > i_1$, dû à l'utilisation d'une décision dure $\hat{u}[\mathcal{E}]_0^{i_1}$ inversée par rapport à celle du SC. On parlera donc d'un *chemin* (ou *trajectoire*) de décodage différent. Le résultat d'un tel décodage est le vecteur $\hat{u}[\mathcal{E}]_0^{N-1}$ estimation de la séquence u_0^{N-1}. Pour simplifier les notations lorsqu'aucune confusion n'est possible, le vecteur $\hat{u}[\mathcal{E}]_0^{N-1}$ sera simplement noté $\hat{u}_0^{N-1}$. Soulignons que si $\mathcal{E} = \emptyset$, le décodage est exactement celui du SC.

Puisque tout chemin dans l'arbre des possibilités peut être décrit par une inversion binaire, il existe, pour toute séquence d'information u_0^{N-1} et toute séquence reçue $Y = y_0^{N-1}$, une unique inversion binaire $\mathcal{E}_Y$ telle que $\hat{u}[\mathcal{E}_Y]_0^{N-1} = u_0^{N-1}$ ($\mathcal{E}_Y$ peut être calculé à l'aide d'un décodeur genie-aided, tel qu'expliqué

Algorithme 4 Décodage par inversion d'un code polaire $\mathcal{C}(N,K,\mathcal{I})$

procédure SC(y_0^{N-1}, T, $\mathcal{L}_{\text{flip}} = \{\mathcal{E}_1,\ldots,\mathcal{E}_T\}$)
  $\hat{u}_0^{N-1} \leftarrow SC(\emptyset)$
  si CRC($\hat{u}_0^{N-1}$) correct **alors** retourner $\hat{u}_0^{N-1}$; **fin si**
  pour $t = 1,\ldots,T$ **faire**
    $\hat{u}_0^{N-1} \leftarrow$SC($\mathcal{E}_t$);
    si CRC($\hat{u}_0^{N-1}$) correct **alors** retourner $\hat{u}_0^{N-1}$; **fin si**
  fin pour
  retourner $\hat{u}_0^{N-1}$; ▷ Erreur de décodage, car CRC non vérifié
fin procédure

plus loin, à la Section 3.2). La problématique fondamentale du décodage à inversion est d'identifier $\mathcal{E}_Y$ avec une forte probabilité et en un minimum de tentatives de manière à limiter la complexité du décodage.

Une description haut-niveau du décodage par inversion est donnée dans l'algorithme 2. Cet algorithme est muni d'une liste d'inversions binaires, notée $\mathcal{L}_{\text{flip}} = \{\mathcal{E}_1,\ldots,\mathcal{E}_T\}$, décrivant les T tentatives, notées SC($\mathcal{E}_t$), à réaliser tant que le CRC n'est pas vérifié. Deux conditions sont nécessaires pour qu'un tel décodeur obtienne la bonne séquence $\hat{u}_{\mathcal{I}}$ (en supposant que le SC n'y est pas parvenu):

- que la liste $\mathcal{L}_{\text{flip}}$ contienne l'unique inversion binaire $\mathcal{E}_Y$ corrigeant la trajectoire du SC
- qu'aucune tentative précédente ne vérifie le CRC

Dès lors, il est clair que les performances d'un tel décodeur dépendent directement de la méthode utilisée pour construire la liste $\mathcal{L}_{\text{flip}}$, qui doit s'efforcer de maximiser $\Pr(\mathcal{E}_Y \in \mathcal{L}_{\text{flip}})$ et minimiser $\mathbb{E}(t \in \{1,\ldots,T\} \mid \mathcal{E}_t = \mathcal{E}_Y\}$. Cette dernière quantité est également associée au nombre moyen de tentatives effectuées, et sa minimisation est essentielle pour limiter la complexité du décodeur.

Dans la description du décodage à inversion de l'algorithme 4, la liste $\mathcal{L}_{\text{flip}}$ est présentée comme étant une donnée d'entrée du décodeur, laissant entendre qu'elle serait prédéterminée a priori et indépendante de la réalisation de bruit (*i.e.* de la séquence reçue). Une telle approche est envisageable, par exemple, en exploitant les informations relatives à la fiabilité des canaux virtuels de manière à cibler les positions les plus critiques. Néanmoins, il est clair que pour atteindre d'excellentes performances, il convient de construire la liste dynamiquement pour chaque réalisation de bruit en exploitant les valeurs des LLRs obtenus lors des tentatives de décodage. Cela suppose de définir une métrique capable d'évaluer précisément la probabilité qu'une inversion binaire corrige la trajectoire du SC. On verra qu'il est cependant possible de combiner les deux approches en construisant la liste $\mathcal{L}_{\text{flip}}$ dynamiquement à partir d'un nombre réduit d'inversions binaires, les autres étant jugées trop peu probables d'après des critères indépendants de la réalisation de bruit.

Après une première approche simple illustrant l'avantage et la faisabilité de corriger plusieurs erreurs [19], nous avons proposé un algorithme appelé décodage à inversion dynamique (D-SCFlip) dans [20]. Ce chapitre s'inscrit dans le prolongement de ces travaux de recherche. Par rapport au décodeur à inversion proposé dans [3], et appelé SCFlip, l'algorithme D-SCFlip dispose de deux améliorations:

- Dans le SCFlip, seuls les inversions binaires d'ordre $\omega = 1$ sont considérées, mais il s'avère qu'une telle contrainte introduit une limitation préjudiciable des performances (cf section 3.2). L'algorithme proposé est capable de considérer des inversions binaires d'ordre supérieur et offre un gain significatif.
- La métrique utilisée pour générer et trier dynamiquement la liste $\mathcal{L}_{\text{flip}}$ est re-définie de manière à prendre en compte des inversions binaires de tout ordre $\omega \geq 1$. Cette nouvelle métrique permet

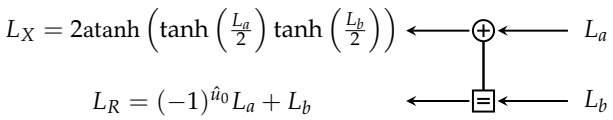


FIGURE 3.1: *Impact de la décision $\hat{u}_0$ sur $\hat{u}_1$ dans le noyau G_2*

également de mieux prendre en compte l'aspect successif du décodage SC et améliore notablement les performances tout en apportant une réduction significative du nombre moyen de tentatives.

La notion fondamentale du décodage à inversion est celle d'inversion binaire d'ordre ω. Imposer une limite sur l'ordre maximal considéré (ou autrement dit sur le nombre maximum d'inversions appliquées) réduit le nombre de trajectoires susceptibles d'être explorées et facilite l'identification de $\mathcal{E}_Y$ lorsque celui-ci en fait partie. Dans le cas contraire, le décodeur tournera dans le vide en cherchant une trajectoire qui a dès le départ été écartée. La question de l'impact d'une limitation de l'ordre des inversions constitue le point de départ incontournable de tout décodeur à inversion.

3.2 BORNES D'ORDRE ω

Rappelons que le décodage SC exploite les estimations dures des bits précédents $\hat{u}_0^{i-1}$ pour le décodage d'un bit $u_i, i \in \mathcal{I}$. Dès lors, toute erreur durant le décodage va se propager et affecter le décodage des bits ultérieurs. Par contre, le simple fait de rétablir l'algorithme sur le bon chemin en inversant la *toute première erreur* permet de compenser toute une série d'erreur dues à la propagation. Pour autant, cela ne suffit pas à garantir un décodage correct, et il peut être nécessaire d'inverser plusieurs décisions. Lorsque l'on parlera d'erreur dans la suite de ce chapitre, il ne sera pas question des erreurs dues à la propagation, mais de celles apparaissant même lorsque tous les bits précédents sont correctement décodés/figés, car seules ces décisions-ci doivent être identifiées et inversées. Dans cette section, on se propose d'étudier l'impact de corriger ω erreurs.

On définit le concept de décodeur à inversion *idéal*, d'ordre ω, un décodeur corrigeant toute réalisation de bruit y_0^{N-1} si et seulement si $|\mathcal{E}_Y| \leq \omega$. En notant $\omega_Y = |\mathcal{E}_Y|$, la performance d'un tel décodeur est définie par:

$$\text{iWER-}\omega = \Pr(\omega_Y > \omega) \tag{3.3}$$

Cette probabilité correspond à la performance optimale qu'un décodeur pratiquant ω inversions est susceptible d'atteindre – sous réserve d'identifier toutes les inversions binaires $\mathcal{E}_Y$ lorsque $|\mathcal{E}_Y| \leq \omega$. Celle-ci peut être efficacement obtenue à l'aide un décodeur *genie-aided* semblable à celui utilisé pour réaliser la construction du code de manière heuristique (cf section 1.3.2). Pour rappel, un tel décodeur calcule les valeurs des LLRs en connaissant les valeurs exactes des bits d'information précédents, puis prend la décision dure $u_i^{(ga)}$ basée sur le signe de ce LLR. Étant donné un ensemble $\mathcal{I}$, l'inversion binaire $\mathcal{E}_Y$ permettant de décoder correctement la réalisation de bruit y_0^{N-1} s'obtient par:

$$\mathcal{E}_Y = \{i \in \mathcal{I}, u_i \neq u_i^{(ga)}\}, \tag{3.4}$$

d'où l'on dérive finalement la valeur $\omega_Y = |\mathcal{E}_Y|$. Bien que cette méthode soit uniquement basée sur des simulations, elle reste à ce jour la seule solution pour dériver ces bornes, dans la mesure où il n'existe pas de formule analytique permettant de les calculer. Voici un exemple simple illustrant la difficulté du problème.

Considérons le noyau G_2 présenté dans la figure 3.1. Ce noyau peut correspondre au cas trivial d'un code $N = 2$, mais il convient plutôt de le considérer comme n'importe quel noyau du dernier niveau dans le sens de décodage (*i.e.*, le niveau le plus à gauche) du graphe d'un code polaire. Pour simplifier,

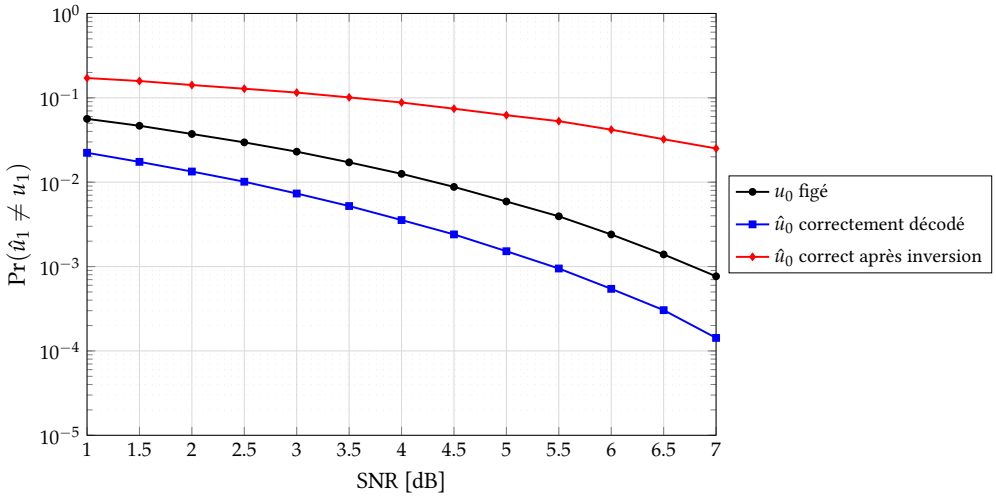


FIGURE 3.2: *Probabilité d'erreur sur $\hat{u}_1$ dans différents cas de figure*

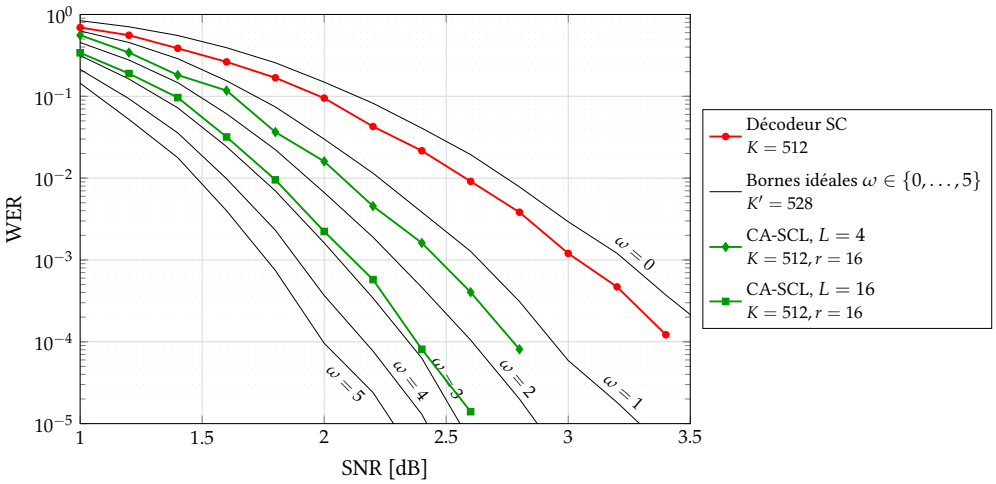


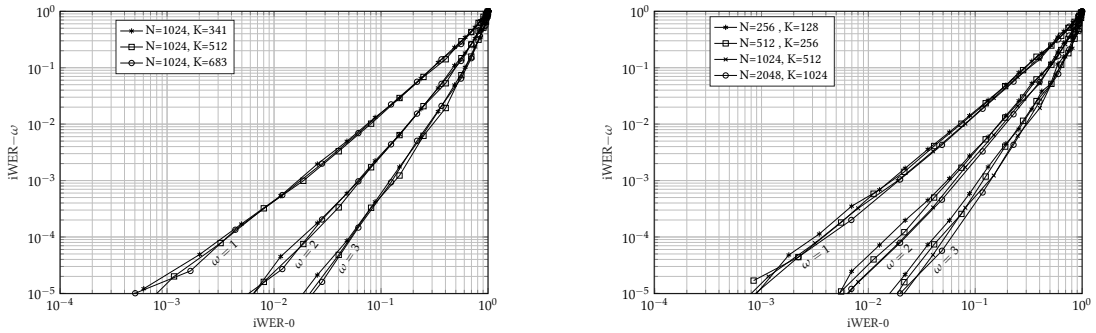
FIGURE 3.3: *Bornes idéales* i *WER-ω d'un décodeur à ω inversions* $(N,K) = (1024,512)$

les notations utilisées resteront celles du noyau et on supposera que $u_0 = u_1 = 0$. On se propose de comparer la probabilité d'erreur sur $\hat{u}_1$ dans les trois cas suivants: le cas où u_0 est un bit figé, celui où $\hat{u}_0$ est correctement décodé par le SC, et celui où $\hat{u}_0$ est correct après inversion (donc incorrectement décodé par le SC). Sachant que $\text{sign}(L_X) = \text{sign}(L_a)\text{sign}(L_b)$, on obtient:

$$\begin{aligned}
\Pr(\hat{u}_1 \neq u_1 \mid u_0) &= \Pr(L_a + L_b < 0) \\
\Pr(\hat{u}_1 \neq u_1 \mid u_0, L_X > 0) &= \Pr(L_a + L_b < 0 | L_a \cdot L_b > 0) \\
&= \Pr(L_a < 0) \cdot \Pr(L_b < 0) \\
\Pr(\hat{u}_1 \neq u_1 \mid u_0, L_X < 0) &= \Pr(L_a + L_b < 0 | L_a \cdot L_b < 0) \\
&= 2\Pr(L_a + L_b < 0 | L_a < 0, L_b > 0)
\end{aligned}$$

Si l'on suppose que L_a et L_b suivent des distributions gaussiennes symétriques, cette dernière probabilité peut être efficacement calculée. Les résultats sont présentés sur la figure 3.2. Il apparaît nettement – et assez logiquement – que le fait qu'une première inversion ait été nécessaire augmente significativement la probabilité qu'une nouvelle erreur survienne par rapport au cas où la décision était correcte dès le décodage SC.

La figure 3.3 présente ces bornes idéales pour un code $(N,K) = (1024,512)$. Celles-ci sont représen-



(a) Impact du rendement

(b) Impact de la longueur du code

FIGURE 3.4: *Bornes idéales en fonction de la longueur et du rendement du code*

tées avec $K' = 528$ par anticipation des bits de CRC nécessaires dans le décodage à inversion. Cela explique la différence entre la courbe rouge (décodeur SC pour $K = 512$ bits) et la courbe correspondant à $\omega = 0$ (décodeur SC pour $K' = 528$ bits) – notons néanmoins que c'est la première des deux qui doit servir de référence. A titre de comparaison, les performances du CA-SCL sont également présentées pour $L = 4$ et $L = 16$, respectivement. Si corriger une erreur apporte déjà un gain significatif (de l'ordre de 0.4dB pour un taux d'erreur de 10^{-3}), parvenir à en corriger 2 voire 3 est nécessaire pour s'approcher du CA-SCL utilisant une taille de liste jusqu'à $L = 16$.

Mentionnons tout de même que la dérivation des bornes iWER-ω est considérablement simplifiée sur un canal BEC, de sorte que des résultats analytiques sont possibles. Dans [66], une borne inférieure est proposée et est utilisée pour démontrer que le *scaling exponent* d'un code polaire avec un décodeur genie-aided reste constant. Dans [20], nous avons proposé une méthode analytique pour calculer précisément ces bornes sur le canal BEC. Celle-ci repose sur l'interprétation d'une réalisation de bruit reçue comme un code polaire poinçonné sur un canal BEC parfait, de sorte que les réalisations de bruit peuvent être regroupées par classes d'équivalence, pouvant être exhaustivement énumérées tant que $N \leq 64$. Le détail de cette méthode est proposé en Annexe C.

Finalement, nous avons observé la variabilité du gain obtenu par un décodeur idéal d'ordre ω par rapport au décodeur SC, pour des codes de longueurs et de rendements différents. Les résultats sont donnés dans la figure 3.4. Il peut être observé que, pour un même iWER-0, le gain reste très similaire pour tous les codes considérés, de sorte que l'on peut estimer que deux ou trois inversions sont suffisantes en général pour obtenir un gain significatif – sous réserve d'identifier efficacement l'inversion binaire correcte.

3.3 MÉTRIQUE POUR LE DÉCODAGE À INVERSION

3.3.1 PROBLÉMATIQUE

Dans [3], la métrique utilisée pour évaluer la probabilité qu'une inversion binaire $\mathcal{E} = \{i\}$ d'ordre $\omega = 1$ corrige la trajectoire du SC est donnée par:

$$M(\{i\}) = |L_i(y_0^{N-1}, \hat{u}_0^{i-1})|, \tag{3.5}$$

et la liste $\mathcal{L}_{\text{flip}}$ est construite une fois pour toute à l'issue du décodage SC avec les T inversions binaires ayant la plus faible métrique. A noter que la métrique (3.5) est équivalente à la version simplifiée de la

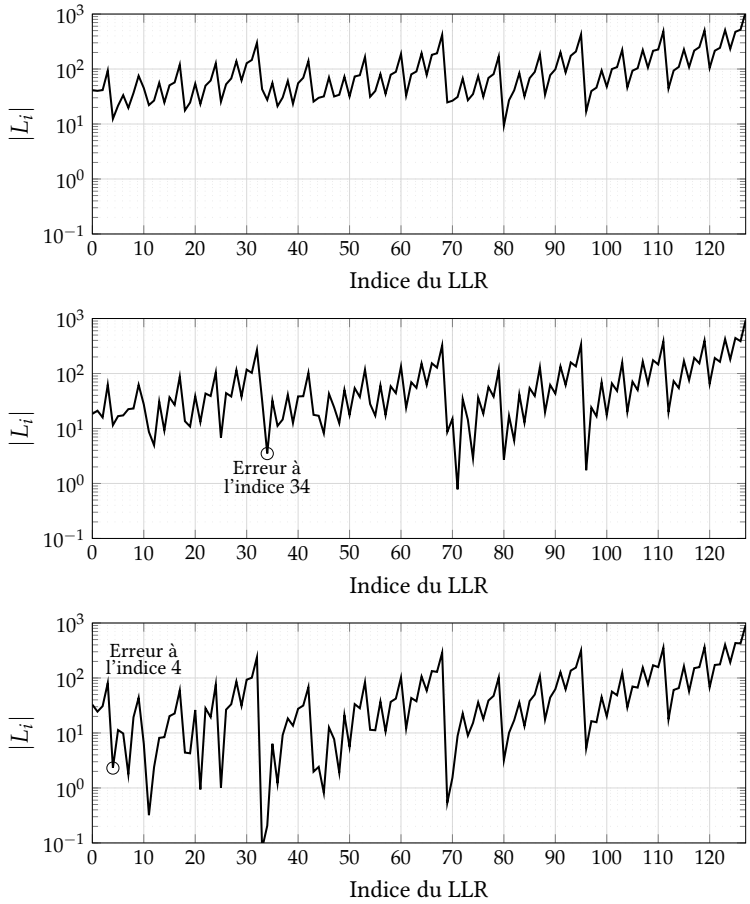


FIGURE 3.5: *Exemple de valeurs de LLRs pour un code* $(N, K) = (256, 128)$

métrique du décodage par liste, mais sans prendre en compte les LLRs des bits figés. Cependant, cette métrique ne parait pas la plus adaptée pour rendre le décodeur à inversion réellement compétitif, et ce pour deux raisons.

Premièrement, la partie précédente a mis en évidence l'importance pour le décodeur à inversion d'être capable de corriger plusieurs erreurs, ce que la métrique ci-dessus ne permet pas. Il faut donc redéfinir une métrique permettant d'évaluer la probabilité d'une inversion binaire d'ordre $\omega > 1$. En second lieu, la métrique (3.5) ne prend pas pleinement en compte le processus successif du décodage. En effet, il est clair que lorsqu'une erreur survient, la probabilité d'erreur sur les bits d'information suivants augmente drastiquement. Ce phénomène d'erreurs en cascade se retrouve au niveau des valeurs absolues des LLRs, qui sont susceptibles d'être significativement plus faibles après une erreur que lorsque la trajectoire est correcte. La figure 3.5 présente trois illustrations de la répartition des valeurs de LLRs à l'issue d'un décodage SC pour un code $(N, K) = (256, 128)$. Le premier cas représente une réalisation de bruit décodée correctement, les deux suivants étant des réalisations en erreur avec la première erreur en position 35 et 4 respectivement. Il peut être observé que lorsque la première erreur survient tôt dans le décodage, celle-ci devient difficile à identifier uniquement à partir de la valeur absolue de son LLR. De fait, la métrique (3.5) ne parait pas optimale pour le décodage à inversion, puisqu'elle n'identifie pas la toute première erreur.

Un autre élément qui entre en ligne de compte pour la définition d'une métrique pour le décodage à inversion est la question de l'utilisation des bits figés. Si le SCL est capable d'en tirer profit en comparant des chemins parallèles, ils restent difficiles à exploiter dans les approches nécessitant la comparaison de trajectoires de longueurs variables, où le nombre de bit figés utilisés dépend de la longueur du chemin considéré. Qui plus est, les bits figés correspondent aux canaux virtuels les moins fiables, ce qui implique

que la quantité d'information à exploiter est réduite. Dans [100], une métrique pour le décodage SCS exploitant les bits figés est proposée mais sans amélioration des performances (seule la complexité est réduite). D'un autre côté, le fait de ne pas utiliser les LLRs des bits figés permet de se passer du calcul des LLRs des bits figés et d'utiliser la même optimisation algorithmique que dans le SSC pour ce qui est des sous-blocs de rendement $R = 0$ (cf section 1.4.5).

Finalement, la question de la complexité de calcul de la métrique entre également en ligne de compte. De la même manière que la métrique du décodeur par liste dispose d'une version simplifiée essentielle pour les implémentations matérielles, la simplicité calculatoire de la métrique proposée sera également discutée.

3.3.2 Métrique proposée

Le point de départ est d'évaluer la probabilité qu'une inversion binaire $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\} \subset \mathcal{I}$ d'ordre ω corrige la trajectoire du SC. Par *corriger la trajectoire*, on entend que la tentative $\mathrm{SC}(\mathcal{E}_\omega)$ décode correctement tous les bits de u_0 à u_{i_ω},mais sans garantie quant à l'exactitude du décodage des bits subséquents $u_{i_\omega+1}^{N-1}$.

Considérons l'inversion binaire $\mathcal{E}_{\omega-1} = \{i_1, \ldots, i_{\omega-1}\} \subset \mathcal{E}_\omega$, et notons $\mathrm{L}[\mathcal{E}_{\omega-1}]_i$ et $\hat{u}[\mathcal{E}_{\omega-1}]_i$ les LLRs et les décisions dures dérivés lors de la tentative $\mathrm{SC}(\mathcal{E}_{\omega-1})$. Si l'on compare les trajectoires $\mathrm{SC}(\mathcal{E}_{\omega-1})$ et $\mathrm{SC}(\mathcal{E}_\omega)$, il est clair que l'on a :

$$\text{(i)} \quad \mathrm{L}[\mathcal{E}_{\omega-1}]_0^{i_\omega} = \mathrm{L}[\mathcal{E}_\omega]_0^{i_\omega} \tag{3.6}$$

$$\text{(ii)} \quad \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = \hat{u}[\mathcal{E}_\omega]_0^{i_\omega-1} \tag{3.7}$$

$$\text{(iii)} \quad \hat{u}[\mathcal{E}_\omega]_{i_\omega} = 1 - \hat{u}[\mathcal{E}_\omega]_{i_\omega} \tag{3.8}$$

La probabilité $P(\mathcal{E}_\omega)$ que l'inversion binaire $\mathcal{E}_\omega$ corrige la trajectoire du SC peut s'exprimer comme suit:

$$\begin{aligned}
P(\mathcal{E}_\omega) &= \Pr(\hat{u}[\mathcal{E}_\omega]_0^{i_\omega} = u_0^{i_\omega} \mid y_0^{N-1}) \\
&= \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_{i_\omega} \neq u_{i_\omega}\,,\, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = u_0^{i_\omega-1} \mid y_0^{N-1}) \\
&= \Pr\left(\hat{u}[\mathcal{E}_{\omega-1}]_{i_\omega} \neq u_{i_\omega} \mid y_0^{N-1},\, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = u_0^{i_\omega-1}\right) \cdot \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = u_0^{i_\omega-1} \mid y_0^{N-1}) \\
&= \Pr\left(\hat{u}[\mathcal{E}_{\omega-1}]_{i_\omega} \neq u_{i_\omega} \mid y_0^{N-1},\, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = u_0^{i_\omega-1}\right) \cdot P(\mathcal{E}_{\omega-1}) \\
&\quad \cdot \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_{i_{\omega-1}+1}^{i_\omega-1} = u_{i_{\omega-1}+1}^{i_\omega-1} \mid y_0^{N-1}, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_{\omega-1}} = u_0^{i_{\omega-1}}) \\
&= \Pr\left(\hat{u}[\mathcal{E}_{\omega-1}]_{i_\omega} \neq u_{i_\omega} \mid y_0^{N-1},\, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i_\omega-1} = u_0^{i_\omega-1}\right) \cdot P(\mathcal{E}_{\omega-1}) \\
&\quad \cdot \prod_{j=i_{\omega-1}+1}^{i_\omega-1} \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_j = u_j \mid y_0^{N-1},\, \hat{u}[\mathcal{E}_{\omega-1}]_0^{j-1} = u_0^{j-1}) \\
&= p_e(\hat{u}[\mathcal{E}_{\omega-1}]_{i_{\omega-1}}) \cdot \prod_{j=i_{\omega-1}+1}^{i_\omega-1} (1 - p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j)) \cdot P(\mathcal{E}_{\omega-1})
\end{aligned}$$

avec $p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) \overset{\text{def}}{=} \Pr\left(\hat{u}[\mathcal{E}_{\omega-1}]_j \neq u_j \mid y_0^{N-1}, \hat{u}[\mathcal{E}_{\omega-1}]_0^{j-1} = u_0^{j-1}\right)$. Dans la mesure où $p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) = 0$ pour les bits figés ($j \notin \mathcal{I}$), ces valeurs pourront ne pas être prises en compte dans le produit $\prod_{j=i_{\omega-1}+1}^{i_\omega-1}(1 - p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j))$ ci-dessus. En raisonnant par récurrence, et en utilisant les équations (3.5)-(3.7) – dans lesquelles ω pourra être remplacé par n'importe quel autre $\omega' < \omega$ – on obtient finalement:

$$P(\mathcal{E}_\omega) = \prod_{j \in \mathcal{E}_\omega} p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) \cdot \prod_{\substack{j < i_\omega \\ j \in \mathcal{I} \setminus \mathcal{E}_\omega}} \left(1 - p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j)\right) \tag{3.9}$$

Calculer $p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j)$ est très délicat dans la mesure où cette probabilité est conditionnelle du fait que les bits précédents ont été correctement décodés lors de SC($\mathcal{E}_{\omega-1}$) (voir aussi l'exemple de la figure 3.5). A la place, il est possible de calculer la probabilité $q_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) \stackrel{\text{def}}{=} \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_j \neq u_j | y_0^{N-1}, \hat{u}[\mathcal{E}_{\omega-1}]_0^{j-1})$, conditionnelle des bits précédemment décodés, mais sans considération sur l'exactitude de leurs valeurs, et donnée par (cela provient directement de la définition de $\mathrm{L}[\mathcal{E}_{\omega-1}]_j$):

$$q_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) = \frac{1}{1+\exp\left(|\mathrm{L}[\mathcal{E}_{\omega-1}]_j|\right)} \qquad \forall j \in \mathcal{I} \tag{3.10}$$

On se propose d'utiliser $q_e(\hat{u}[\mathcal{E}_{\omega-1}]_j)$ comme approximation de $p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j)$, mais en introduisant un paramètre de perturbation noté α. Ce paramètre, dont la valeur doit encore être déterminée, vise à ajuster la valeur de q_e de manière à mieux approximer p_e et ainsi mieux tenir compte de la propagation des erreurs dans le décodage. En pratique, cette valeur peut-être optimisée par simulation Monte-Carlo. En remplaçant $p_e(\hat{u}[\mathcal{E}_{\omega-1}]_j) \approx \frac{1}{1+\exp(\alpha|\mathrm{L}[\mathcal{E}_{\omega-1}]_j|)}$ dans l'équation (3.9), la probabilité $P(\mathcal{E}_\omega)$ devient:

$$P_\alpha(\mathcal{E}_\omega) = \prod_{j\in\mathcal{E}_\omega}\left(\frac{1}{1+\exp\left(\alpha|\mathrm{L}[\mathcal{E}_{\omega-1}]_j)|\right)}\right) \cdot \prod_{\substack{j<i_\omega \\ j\in\mathcal{I}\setminus\mathcal{E}_\omega}}\left(\frac{1}{1+\exp\left(-\alpha|\mathrm{L}[\mathcal{E}_{\omega-1}]_j|\right)}\right) \tag{3.11}$$

En utilisant le fait que $\frac{1}{1+\exp(x)} = \frac{\exp(-x)}{1+\exp(-x)}$, l'équation (3.11) se reformule par:

$$P_\alpha(\mathcal{E}_\omega) = \prod_{j\in\mathcal{E}_\omega}\exp\left(-\alpha|\mathrm{L}[\mathcal{E}_{\omega-1}]_j)|\right) \cdot \prod_{\substack{j\leq i_\omega \\ j\in\mathcal{I}}}\left(\frac{1}{1+\exp\left(-\alpha|\mathrm{L}[\mathcal{E}_{\omega-1}]_j|\right)}\right) \tag{3.12}$$

Finalement, la métrique proposée s'obtient en prenant le logarithme de cette quantité:

Définition 8 *La métrique associée à une inversion binaire $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\} \subset \mathcal{I}$, d'ordre ω, est donnée par:*

$$\begin{aligned} M_\alpha(\mathcal{E}_\omega) &= \sum_{j\in\mathcal{E}_\omega}|L[\mathcal{E}_{\omega-1}]_j| + \sum_{\substack{j<i_\omega \\ j\in\mathcal{I}}}\psi_\alpha(|\,L[\mathcal{E}_{\omega-1}]_j\,|) \\ \text{où } \psi_\alpha(x) &= \frac{1}{\alpha}\log(1+\exp(-\alpha\cdot|x|)) \end{aligned} \tag{3.13}$$

Les inversions binaires ayant la plus forte probabilité de corriger la trajectoire sont ceux minimisant cette métrique. La métrique d'une inversion binaire $\mathcal{E}_\omega$ peut se calculer à partir de celle de l'inversion binaire $\mathcal{E}_{\omega-1} \subset \mathcal{E}_\omega$ et des LLRs obtenus lors de la tentative SC($\mathcal{E}_{\omega-1}$) par:

$$M_\alpha(\mathcal{E}_\omega) = M_\alpha(\mathcal{E}_{\omega-1}) + |\mathrm{L}[\mathcal{E}_{\omega-1}]_{i_\omega}| + \sum_{\substack{j=i_{\omega-1}+1 \\ j\in\mathcal{I}}}^{i_\omega-1}\psi_\alpha(|\,\mathrm{L}[\mathcal{E}_{\omega-1}]_j\,|) \tag{3.14}$$

Pour une inversion binaire $\mathcal{E}_1 = \{i_1\}$ d'ordre $\omega = 1$, la métrique s'écrit:

$$M_\alpha(\mathcal{E}_1) = |\mathrm{L}_{i_1}| + \sum_{\substack{j<i_1 \\ j\in\mathcal{I}}}\psi_\alpha(|\,\mathrm{L}_j\,|), \tag{3.15}$$

où $\mathrm{L}_j, j \in \mathcal{I}$ sont les LLRs calculés lors du décodage SC. Par rapport à la métrique (3.5), le terme supplémentaire permet de prendre en compte l'aspect successif du décodeur, puisqu'une inversion binaire sera pénalisé au niveau de sa métrique lorsque des LLRs faibles surviennent plus tôt dans le décodage.

Pour évaluer l'impact du coefficient α, considérons les deux cas limites:

- Quand $\alpha \to +\infty$, $\lim_{\alpha\to+\infty} \psi_\alpha(x) = 0$, de sorte que $M_\infty(\mathcal{E}_\omega) = \sum_{j\in\mathcal{E}_\omega} |\mathrm{L}[\mathcal{E}_{\omega-1}]_j|$ est la somme des valeurs absolues des LLRs des positions inversées. Dans le cas particulier d'une inversion binaire d'ordre $\omega = 1$, on retrouve la métrique (3.5). Ce cas revient à ne pas tenir compte de l'aspect successif du décodage.
- Quand $\alpha = 0$, on peut remarquer que $P_0(\mathcal{E}_\omega) = \left(\frac{1}{2}\right)^{k_\mathcal{I}}$, où $k_\mathcal{I}$ correspond au nombre d'indices de $\mathcal{I}$ inférieurs ou égaux à i_ω, de sorte que les inversions binaires sont triées d'après la valeur du dernier indice de $\mathcal{E}_\omega$.
- Pour $\alpha \in]0,+\infty[$, la valeur de α décrit un compromis entre privilégier les bits ayant les plus faibles LLRs en valeur absolue et donner la priorité à ceux situés tôt dans le décodage. Remarquons que pour $\alpha = 1$, la métrique est équivalente à la version exacte de celle du décodage par liste, mais sans tenir compte des bits figés.

L'impact du coefficient α peut être visualisé en observant la capacité de la métrique à détecter la première erreur. Pour cela, on utilise un décodeur genie-aided permettant d'identifier l'ordre ω_Y de la réalisation de bruit et, s'il y a lieu, de la position de la première erreur. On génère alors la liste de toutes les inversions binaires d'ordre $\omega = 1$ et l'on détermine la position moyenne de cette première erreur dans la liste, en supposant la liste ordonnée d'après la métrique M_α. Les résultats sont fournis sur la figure 3.6 pour un code polaire $(N,K) = (1024,512)$. On peut observer qu'un minimum se dessine nettement, entre 0.2 et 0.5 suivant la valeur du SNR, et qu'il y a un avantage significatif à utiliser la valeur optimale de α plutôt que $\alpha \to +\infty$ ou $\alpha = 1$. Il faut souligner que la valeur optimale du coefficient α dépend du SNR, de sorte que son optimisation doit être effectuée pour chaque canal.

Néanmoins, il s'avère que la valeur optimale du coefficient α pour un code $\mathcal{C}(N,K)$ à un SNR donné est fortement corrélée au taux d'erreur du décodeur SC pour le code et le canal considéré. Nous avons en effet observé la valeur de α optimisée par simulation Monte-Carlo sur le canal AWGN pour des codes de longueurs $N = \{2^8,2^9,2^{10}\}$ avec trois valeurs de rendements $R \in \{\frac{3}{8},\frac{1}{2},\frac{4}{8}\}$ et des valeurs de SNR telles que le taux d'erreur du SC varie de 10^{-1} à 10^{-4}. Les résultats sont présentés dans la figure 3.7 et montrent que la valeur optimal de α est fortement corrélée au taux d'erreur du SC, de sorte qu'un modèle empirique peut être obtenu :

$$\alpha_m(\mathrm{WER_{SC}}) = a_1 \cdot \log(\mathrm{WER_{SC}})^2 + a_2 \cdot \log(\mathrm{WER_{SC}}) + a_3, \tag{3.16}$$

avec $a_1 = 0.0038$, $a_2 = 0.0779$ et $a_3 = 0.5716$.

Il s'avère que la métrique proposée fait intervenir des calculs complexes, peu adaptés à une implémentation matérielle. Utiliser la même simplification que celle du SCL ramène à la métrique (3.5), sans prise en compte l'aspect successif du décodage. C'est pourquoi nous envisagerons également de remplacer la fonction ψ_α par la fonction $\overline{\psi}$ réduite à deux segments:

$$\overline{\psi}(x) = \begin{cases} 0 & \text{si } x \geq \theta \\ \overline{\psi}_0 \cdot (1 - x/\theta) & \text{si } 0 \leq x < \theta \end{cases} \tag{3.17}$$

où $\overline{\psi}_0$ et θ sont à déterminer par simulation. La valeur θ correspond à un seuil permettant de déterminer si les métriques ultérieures doivent être pénalisées ou non, au regard des valeurs des LLRs des bits précédents.

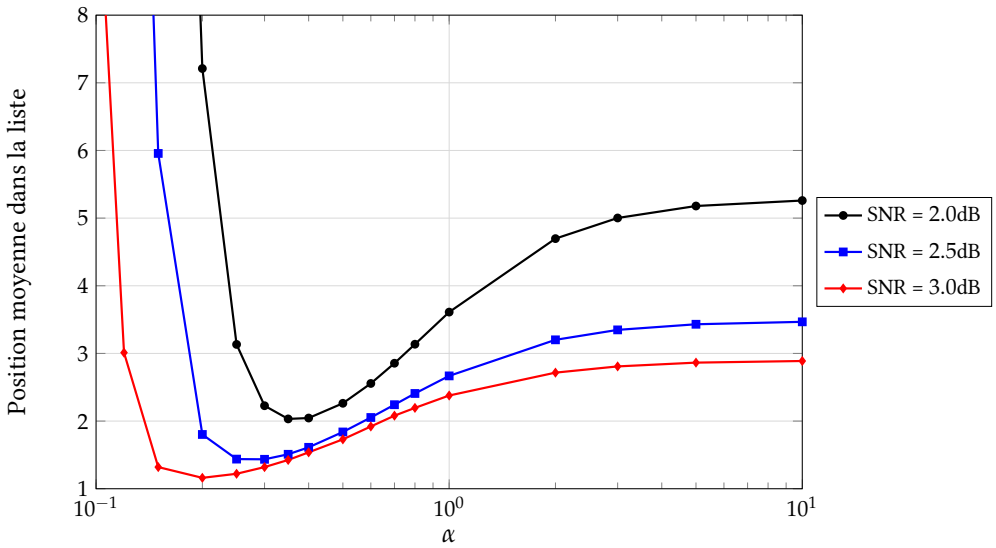


FIGURE 3.6: *Impact du coefficient α sur l'identification de la première erreur pour un code $(N,K) = (1024,512)$*

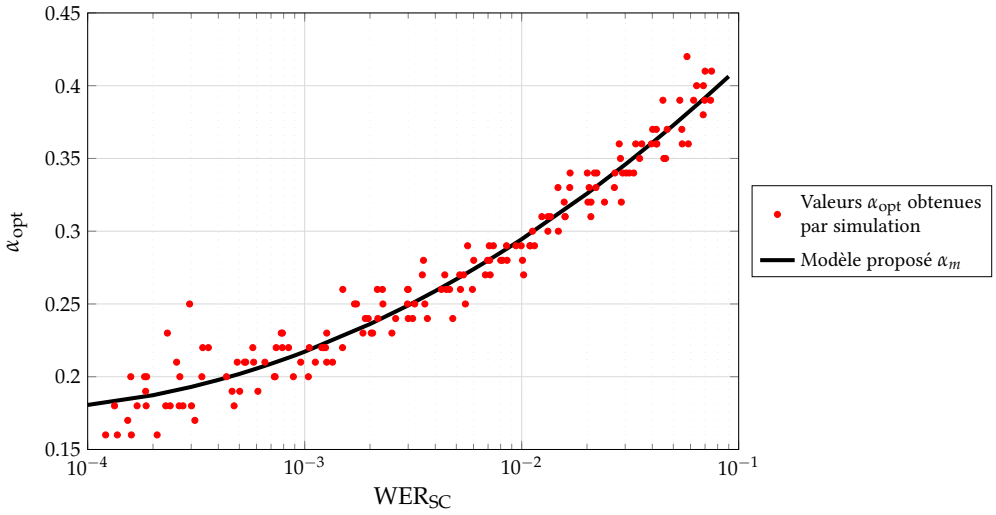


FIGURE 3.7: *Valeur optimale de α sur le canal AWGN en fonction du taux d'erreur du SC. Les points de simulations correspondent à des codes de longueurs et rendements variables, mais la valeur optimale du coefficient reste similaire pour un* WER$_{SC}$ *donné.*

3.3.3 COMPARAISON DES MÉTRIQUES

L'objectif de cette section est de comparer la métrique proposée avec celles de l'état de l'art, en observant la capacité des différentes métriques à identifier la toute première erreur. Pour cela, on génère des réalisations de bruits y_0^{N-1} et l'on détermine, à l'aide d'un décodeur genie-aided, l'inversion binaire $\mathcal{E}_Y = \{i_1,\dots,i_\omega\}$. A chaque fois que $|\mathcal{E}_Y| > 0$, on procède au décodage SC puis l'on génère la liste $\mathcal{L}_{\text{flip}}$ de toutes les inversions binaires d'ordre 1, ordonnée par la métrique considérée. On peut alors déterminer la position de i_1 dans cette liste. Finalement, on comparera l'efficacité des métriques en observant la probabilité que i_1 ne soit pas dans les T premières positions de la liste $f(T) = \Pr(i_1 \notin \mathcal{L}_{\text{flip}}{}^{(1:T)})$. Ce protocole est reproduit à l'identique pour chaque métrique.

Les résultats sont présentés dans la figure 3.8 pour un code $(N,K) = (1024,512)$ à SNR= 2.0dB et pour les métriques suivantes:

(a) La métrique du SCFlip [3] (version simplifiée de la métrique du SCL, sans tenir compte des bits figés)

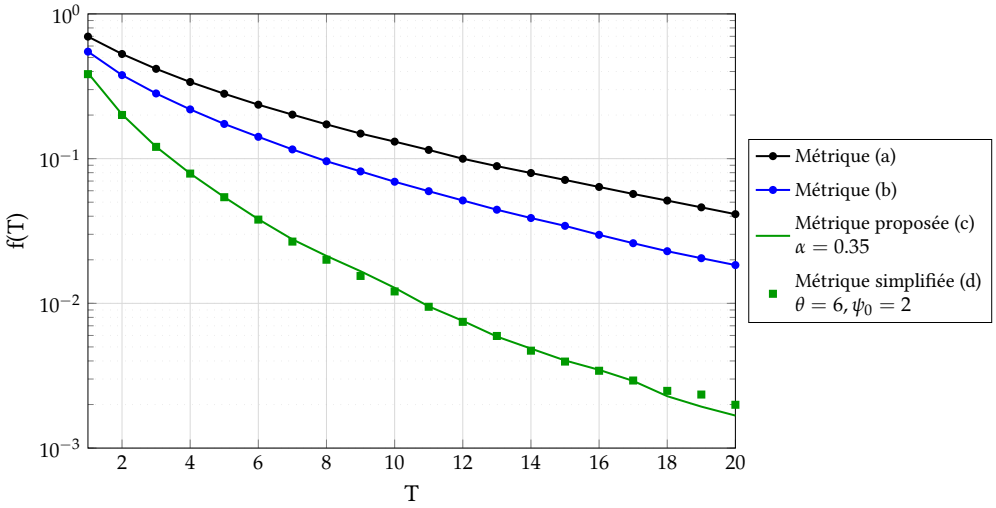


FIGURE 3.8: *Comparaisons des métriques pour* $(N,K) = (1024,512)$ *avec SNR*$= 2.0dB$

(b) La métrique du SCL exacte, sans tenir compte des bits figés (utilisée dans le SCS [72] et équivalente à la métrique proposée pour $\alpha = 1$)

(c) La métrique proposée avec α optimisé par simulation

(d) La métrique simplifiée avec θ et $\psi(0)$ optimisés par simulation

Les résultats mettent en évidence la nette supériorité de la métrique proposée sur les métriques de l'état de l'art: si l'on se donne comme objectif une certaine valeur de $f(T)$, la longueur de la liste T (et donc le nombre maximum de tentatives) pourra être divisée par deux par rapport aux autres métriques. De plus, il apparaît que la formule simplifiée proposée fait tout aussi bien que la métrique exacte.

Si la détection rapide de la toute première erreur est fondamentale pour le décodage à inversion, cela ne garantit pas pour autant de permettre une comparaison efficace pour des inversions binaires d'ordre différents. La partie suivante se chargera de décrire un processus de décodage à plusieurs inversions et les simulations serviront à valider l'efficacité de la métrique pour plusieurs inversions.

3.4 Décodage à inversion dynamique

Cette section décrit un algorithme de décodage à inversion basé sur la métrique proposée, capable de corriger plusieurs erreurs et appelé décodeur à inversions dynamiques D-SCFlip. La principale caractéristique de cet algorithme est la garantie d'explorer les trajectoires par probabilités de succès décroissantes (*i.e.* par métriques croissantes) et permet ainsi de minimiser le nombre moyen de tentatives. Pour cela, la liste $\mathcal{L}_{\text{flip}}$, de longueur T, est construite au fur et à mesure des tentatives, en insérant dans la liste les inversions binaires dont la probabilité est suffisamment élevée. Pour rappel, il est nécessaire de disposer des valeurs de LLRs relatives à la trajectoire $\text{SC}(\mathcal{E}_{\omega-1})$ pour déterminer la métrique de l'inversion binaire $\text{SC}(\mathcal{E}_{\omega})$. Ainsi chaque tentative associée à une inversion binaire $\mathcal{E}_{\omega-1} = \{i_1,\ldots,i_{\omega-1}\}$ permet d'évaluer les métriques des inversions binaires d'ordre immédiatement supérieur $\mathcal{E}_{\omega} = (\mathcal{E}_{\omega-1} \cup \{i\})$, avec $i > i_{\omega-1}, i \in \mathcal{I}$. L'algorithme 5 décrit le processus proposé avec la fonction de mise à jour de la liste dans l'algorithme 6, où $\mathcal{M}_{\text{flip}}$ est la liste des métriques associées à chaque inversion binaire de $\mathcal{L}_{\text{flip}}$. La liste $\mathcal{L}_{\text{flip}}$ est initialisée à l'issue du décodage SC, par les T inversions binaires d'ordre 1 minimisant la métrique M_α (dans l'ordre), tandis que $\mathcal{M}_{\text{flip}}$ est initialisée avec les valeurs des métriques associées à chaque inversion binaire de la liste $\mathcal{L}_{\text{flip}}$.

Il peut être montré qu'un tel algorithme explore les inversions binaires par métrique croissante.

Algorithme 5 Décodeur D-SCFlip

procédure D-SCFLIP($\mathbf{Y}, \mathcal{I}, T$)
  $\left(\hat{u}_0^{N-1}, \{\mathrm{L}_i\}_{i\in\mathcal{I}}\right) \leftarrow$SC($\emptyset$)
  si CRC($\hat{u}_0^{N-1}$) correct **alors**
    retourner $\hat{u}_0^{N-1}$;
  sinon
    Init($\mathcal{L}_{\text{flip}}, \mathcal{M}_{\text{flip}}, \{\mathrm{L}_i\}_{i\in\mathcal{I}}$);
  fin si
  pour $t = 1, \ldots, T$ **faire**
    $\mathcal{E}_t = \mathcal{L}_{\text{flip}}[t]$;
    $\left(\hat{u}_0^{N-1}, \{\mathrm{L}[\mathcal{E}_t]_i\}_{i\in\mathcal{I}}\right) \leftarrow$SC($\mathcal{E}_t$);
    si CRC($\hat{u}_0^{N-1}$) correct **alors**
      retourner $\hat{u}_0^{N-1}$;
    sinon
      MAJ-LISTE($\mathcal{L}_{\text{flip}}, \mathcal{M}_{\text{flip}}, \{\mathrm{L}[\mathcal{E}_t]_i\}_{i\in\mathcal{I}}, \mathcal{E}_t$);
    fin si
  fin pour
  retourner $\hat{u}_0^{N-1}$;
fin procédure

Algorithme 6 Mise à jour de la liste

procédure MAJ-LISTE($\mathcal{L}_{\text{flip}}, \mathcal{M}_{\text{flip}}, \{\mathrm{L}[\mathcal{E}_t]_i\}_{i\in\mathcal{I}}, \mathcal{E}_t$)
  pour $i = \max(\mathcal{E}_t) + 1, \ldots, N-1$ et $i \in \mathcal{I}$ **faire**
    $\mathcal{E} = \mathcal{E}_t \cup \{i\}$;
    Calculer $M_\alpha(\mathcal{E})$;
    si $M_\alpha(\mathcal{E}) < \max(\mathcal{M}_{\text{flip}})$ **alors**
      Insérer $\mathcal{E}$ dans $\mathcal{L}_{\text{flip}}$ et $M_\alpha(\mathcal{E})$ dans $\mathcal{M}_{\text{flip}}$;
    fin si
  fin pour
  retourner $(\mathcal{L}_{\text{flip}}, \mathcal{M}_{\text{flip}})$;
fin procédure

Proposition 21 *Soit $T' \leq T$, le nombre de tentatives effectuées avant l'arrêt du décodage. Alors les T' premières inversions binaires de $\mathcal{L}_{flip}$ sont celles ayant la plus faible métrique parmi toutes les inversions binaires.*

Pour démontrer cette propriété, il suffit de montrer que pour toute inversion binaire $\mathcal{E} = \{i_1, \ldots, i_\omega\}$ d'ordre ω et tel que $M_\alpha(\mathcal{E}) < M_\alpha(\mathcal{E}_{T'})$, alors $\mathcal{E} \in \mathcal{L}_{\text{flip}}$, où $\mathcal{E}_{T'}$ est la dernière inversion binaire dans la liste, au moment où le décodage s'arrête. On procède par induction sur ω. Pour $\omega = 1$, le résultat est une conséquence directe du fait que la liste est initialisée avec les $T \geq T'$ inversions binaires minimisant la métrique. Pour $\omega > 1$, considérons $\mathcal{E}' = \{i_1, \ldots, i_{\omega-1}\} \subset \mathcal{E}$. Puisque $M_\alpha(\mathcal{E}') < M_\alpha(\mathcal{E}) < M_\alpha(\mathcal{E}_{T'})$, l'hypothèse de récurrence impose que $\mathcal{E}' \in \mathcal{L}_{\text{flip}}$. Dès lors, la tentative SC($\mathcal{E}'$) est nécessairement effectuée avant SC($\mathcal{E}_{T'}$). Lors de la mise à jour de la liste à l'issue de SC($\mathcal{E}'$), l'inversion binaire $\mathcal{E}$ est insérée dans la liste $\mathcal{L}_{\text{flip}}$ à une position qui précède nécessairement l'inversion binaire $\mathcal{E}_{T'}$. D'où $\mathcal{E} \in \mathcal{L}_{\text{flip}}$.

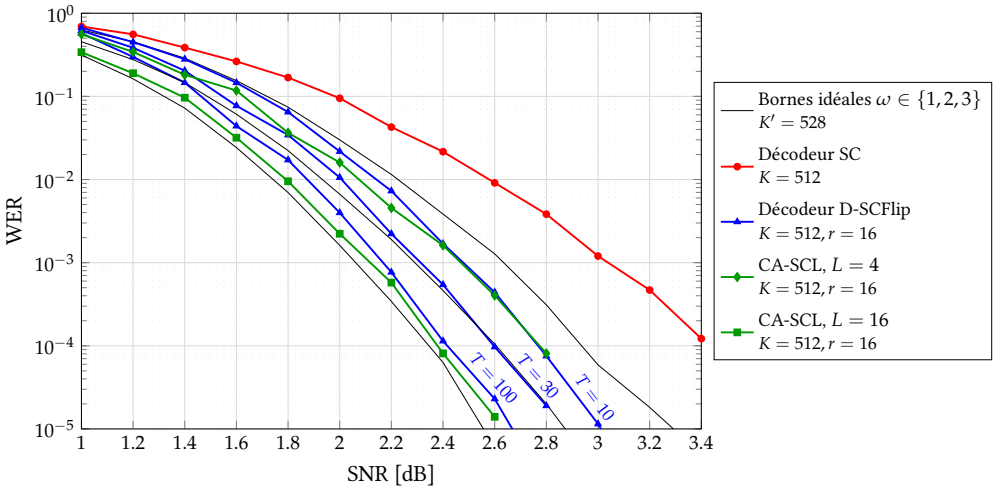


FIGURE 3.9: *Performance du décodeur D-SCFlip pour un code* $(N = 1024, K = 512)$ *avec* $T = 10$, $T = 30$ *et* $T = 100$

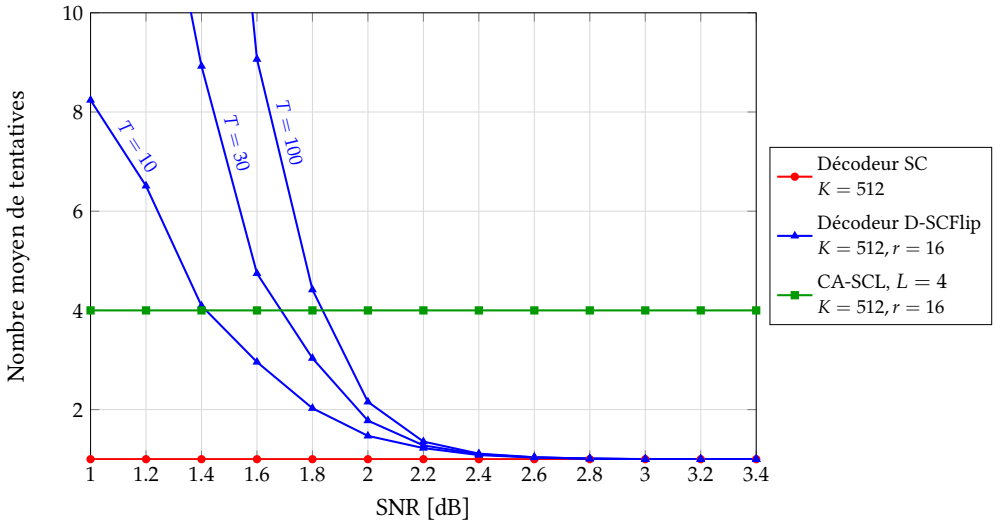


FIGURE 3.10: *Complexité moyenne du décodeur D-SCFlip pour un code* $(N = 1024, K = 512)$ *avec* $T = 10$, $T = 30$ *et* $T = 100$

Dans l'algorithme proposé, il n'y a pas de restriction sur l'ordre des inversions binaires, de sorte que le décodeur D-SCFlip n'est pas contraint par les bornes idéales dérivées dans la section 3.2. Pour autant, il est possible, pour des raisons de complexité ou de facilité d'implémentation, d'en imposer une. Dans ce cas, l'algorithme ne calculera les métriques que pour les inversions binaires d'ordre $\omega \leq \omega_{\max}$.

Les performances du décodeur D-SCFlip sont données sur la figure 3.9, tandis que la figure 3.10 donne le nombre moyen de tentatives, pour un code$(N, K) = (1024, 512)$ avec un CRC de $r = 16$ bits. La valeur optimale du coefficient α est utilisée pour chaque SNR. Il peut être observé que le décodeur proposé parvient à corriger les réalisations de bruit avec une erreur en quelques tentatives et peut même corriger jusqu'à trois erreurs si le nombre de tentatives monte à $T = 100$. Le D-SCFlip avec $T = 100$ est capable de concurrencer le CA-SCL avec $L = 16$, mais dispose d'une complexité moyenne très inférieure pour des SNR modérés et élevés, et tendant rapidement vers celle du décodage SC. La figure 3.11 compare les performances des décodeurs D-SCFlip, CA-SCL et SCFlip [3] pour une longueur de code $N = 256$, des rendements $R \in \{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$ et un CRC de $R = 16$ bits (sauf pour le SC). Il peut être observé que le décodeur D-SCFlip avec $T = 50$ concurrence le CA-SCL avec $L = 8$ pour chaque valeur de rendement.

Les simulations présentées utilisent la métrique définie par l'équation (3.13). Nous avons également

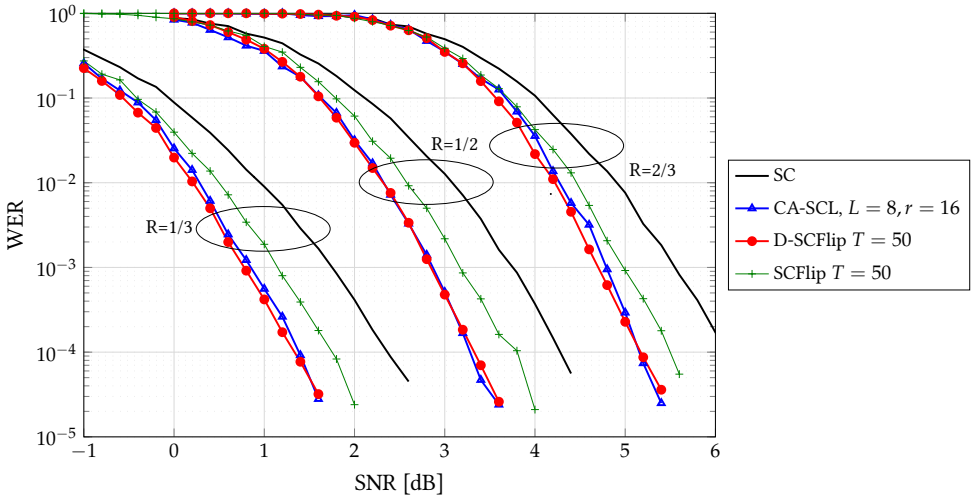


FIGURE 3.11: *Performance du D-SCFlip pour $N = 256$ et différents rendements*

observé l'impact de l'utilisation de la métrique utilisant la fonction de pénalité simplifiée $\overline{\psi}(x)$ de l'équation (3.17). Les résultats montrent que, sous réserve d'une optimisation précise des coefficients θ et $\overline{\psi}_0$, les performances sont tout à fait similaires avec la métrique non simplifiée.

3.5 RÉDUCTION DE COMPLEXITÉ DU DÉCODAGE À INVERSION

3.5.1 DÉCODAGE À INVERSION SIMPLIFIÉ

Il a été expliqué que le décodeur SC a une latence relativement élevée, mais que la solution résidait dans le SSC et l'utilisation d'optimisations algorithmiques permettant de réduire significativement le nombre de calculs de LLRs, notamment pour les sous-blocs de rendement 0 et 1. Puisque le décodage à inversion ne prend pas en compte les LLRs des bits figés, la simplification relative aux nœuds de rendement $R = 0$ est directement applicable. Concernant les nœuds de rendement $R = 1$, le problème est plus délicat dans la mesure où les LLRs sont nécessaires pour calculer les métriques des inversions binaires. Il a été proposé que le SCFlip n'opère le décodage souple des nœuds $R = 1$ que pour la première tentative (celle du SC), de manière à calculer les métriques pour les inversions binaires d'ordre $\omega = 1$ [37]. Cependant, cette solution n'est pas applicable avec un décodeur à plusieurs inversions comme le D-SCFlip. Cette partie développe une solution alternative, inspirée par les solutions mises en place pour le décodage par liste, pleinement compatible avec des inversions binaires d'ordres $\omega > 1$ et avec un impact minime sur les performances. Elle repose sur la re-définition du concept d'inversion binaire, de sorte à pratiquer les inversions directement à l'entrée des nœuds de rendement $R = 1$ [1].

Supposons un code polaire $(N, K, \mathcal{I})$ contenant un sous-bloc de rendement $R = 1$ sur les indices $\{i_m, i_m + 1, \ldots, i_m + 2^{n'}\}$ et de longueur $N' = 2^{n'}$. L'encodage du sous-bloc est donné par:

$$v_{i_m}^{i_m+2^{n'}} = u_{i_m}^{i_m+2^{n'}} \cdot G_{N'} \tag{3.18}$$

Dans le SSC, l'optimisation du décodage d'un tel nœud consiste simplement à déterminer les $\hat{v}_{i_m}^{i_m+2^{n'}}$ à

[1] On utilise ici le sens du décodage de sorte que la sortie d'un bloc de rendement $R = 1$ correspond aux estimations des bits d'information.

partir du signe de leurs LLRs:

$$L(v_{i_m+k}, y_0^{N-1}, \hat{u}_0^{i_m-1}) = \log\left(\frac{\Pr(v_{i_m+k}=0\,|\,y_0^{N-1}, \hat{u}_0^{i_m-1})}{\Pr(v_{i_m+k}=1\,|\,y_0^{N-1}, \hat{u}_0^{i_m-1})}\right) \tag{3.19}$$

puis à obtenir $\hat{u}_{i_m}^{i_m+2^{n'}}$ d'après (3.18). Définissons maintenant un décodeur à inversion simplifié SSCF, en redéfinissant le concept d'inversion binaire de manière à ce que les inversions ne soient plus pratiquées au niveau des $\hat{u}_i$ mais directement à l'intérieur du graphe au niveau des $\hat{v}_i$.

Soit u_0^{N-1}, on note par v_0^{N-1} la séquence obtenue en procédant aux encodage partiels pour les blocs de rendement $R=1$ d'après l'équation (3.18), par $\hat{v}_i$ l'estimation du bit v_i obtenue à partir du signe de son LLR, et par $v_i^{(ga)}$ la décision genie-aided prise d'après le signe du LLR calculé lorsque les décisions précédentes sont connues. Notons que pour un bit d'information n'appartenant pas à un bloc de rendement 1, on aura simplement $u_i = v_i$ et $\hat{u}_i = \hat{v}_i$. On définit alors l'inversion binaire $\mathcal{E}_Y^{(S)}$ par:

$$\mathcal{E}_Y^{(S)} = \{i \in \mathcal{I}, v_i \neq v_i^{(ga)}\}, \tag{3.20}$$

et le nombre d'inversions nécessaires pour décoder correctement y_0^{N-1} est donné par $\omega_Y^{(S)} = |\mathcal{E}_Y^{(S)}|$. Il convient de noter que cette redéfinition de l'ordre d'une réalisation de bruit ne garantit pas nécessairement d'avoir l'égalité entre $\omega_Y^{(S)}$ et ω_Y, de sorte que les bornes idéales sont susceptibles d'évoluer. Si l'on note $\mathcal{B} = \{i_m, \ldots, i_m + 2^{n'}\}$ un nœud de rendement $R=1$ et $\omega_Y[\mathcal{B}]$ et $\omega_Y^{(S)}[\mathcal{B}]$ les ordres de la réalisation de bruit pour le bloc $\mathcal{B}$ dans le cas standard et simplifié respectivement, alors les deux cas prépondérants sont donnés par:

- $\omega_Y^{(S)}[\mathcal{B}] = 0 \Leftrightarrow \omega_Y[\mathcal{B}] = 0$
- s'il n'y a qu'une seule erreur en entrée de $\mathcal{B}$ et que cette erreur correspond au minimum des valeurs absolues des LLRs d'entrée du bloc, alors:

$$\omega_Y^{(S)}[\mathcal{B}] = \omega_Y[\mathcal{B}] = 1$$

Les simulations montrent que les bornes idéales basées sur la définition ci-dessus et celles définies dans la partie 3.2 sont rigoureusement confondues.

Il convient maintenant de redéfinir la métrique associée à la nouvelle définition des inversions binaires. Celle-ci ne se calcule plus à partir des LLRs des bits u_i, mais directement à partir de ceux des v_i pour $i \in \mathcal{I}$. On supposera que le code ne contient qu'un bloc de rendement 1 sur les indices $\{i_m, i_m+1, \ldots i_m+2^{n'}\}$, sans perte de généralité. Supposons une tentative $\mathcal{E} = \{i_1, \ldots, i_k\}$, la métrique associée à l'inversion binaire $\mathcal{E}' = \mathcal{E} \cup \{i_{k+1}\}$ est définie par:

$$M_\alpha^{(S)}(\mathcal{E}') = \begin{cases} M_\alpha(\mathcal{E}') & \text{si } i_{k+1} < i_m \text{ ou } i_{k+1} > i_{m+2^{n'}} \\ \displaystyle\sum_{j\in\mathcal{E}'} |\mathrm{L}[\mathcal{E}]_j| + \sum_{\substack{j<i_m \\ j\in\mathcal{I}}} \psi_\alpha(|\,\mathrm{L}[\mathcal{E}]_j\,|) & \text{sinon.} \end{cases} \tag{3.21}$$

La seule différence avec le cas standard est que les décisions $\hat{v}_{i_m}, \ldots, \hat{v}_{i_m+2^{n'}}$ ne dépendent que des décisions $\hat{v}_0^{i_m-1}$ mais sont indépendantes entre elles. Dès lors, le fait que l'une de ces décisions soit associée à un LLR faible en valeurs absolue ne doit pas pénaliser la métrique des autres. Par contre, pour les métriques calculées après le bloc de rendement 1, la pénalité doit bien prendre en compte tous les LLRs du bloc.

Les simulations sont données dans la figure 3.12, confirmant que cette re-définition des inversions binaires n'a qu'un coût négligeable sur les performances, tout en permettant d'éviter les calculs de LLRs

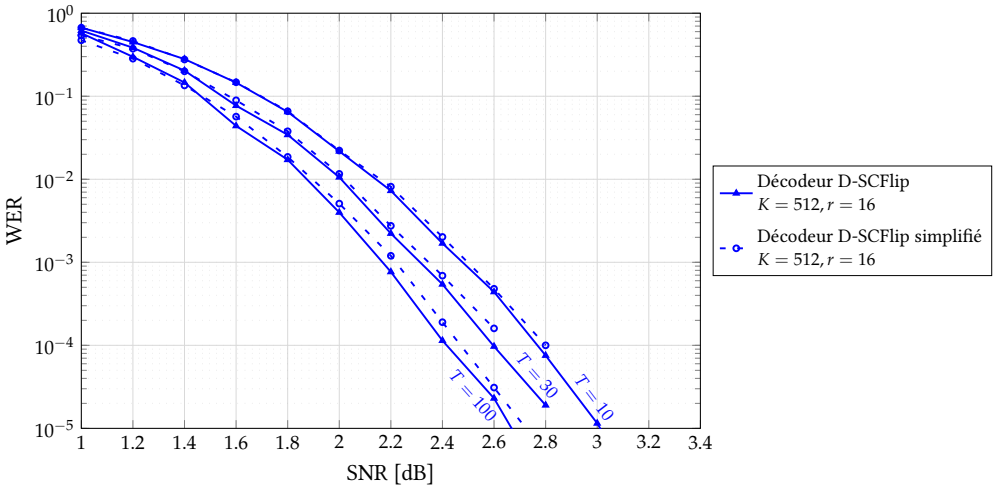


FIGURE 3.12: *Comparaison des décodeurs D-SCFlip standard et simplifié pour un code* $(N = 1024, K = 512)$ *avec* $T = 10$, $T = 30$ *et* $T = 100$

dans les nœuds de rendement $R = 1$. Sur le principe, on peut dire que le décodeur SSCF proposé combine un décodeur à inversion, où les inversions sont pratiquées sur les bits d'information, et un décodeur de Chase où les inversions sont pratiquées sur les bits codés des nœuds de rendement $R = 1$, et où une unique métrique permet d'articuler les deux stratégies entre elles.

3.5.2 RÉDUCTION DES CALCULS DE MÉTRIQUE

L'utilisation du décodage de Chase pour les nœuds de rendement $R = 1$ a été appliquée au décodage par liste afin d'optimiser la latence [83]. Dans [83], une observation empirique a été utilisée afin de réduire le nombre de chemin à explorer. Il est indiqué que pour un bloc $\mathcal{B} = \{i_m, \ldots, i_m + N'\}$, tel que:

$$|L_{j_1}| \leq |L_{j_2}| \leq \cdots \leq |L_{j_{N'}}|, j_k \in \mathcal{B}, \tag{3.22}$$

alors il est suffisant de ne considérer que les quatre inversions binaires suivantes (où $\mathcal{E}[\mathcal{B}] = \mathcal{E} \cap \mathcal{B}$):

1. Le chemin sans inversion ($\mathcal{E}[\mathcal{B}] = \emptyset$)
2. Une inversion sur j_1 ($\mathcal{E}[\mathcal{B}] = \{j_1\}$)
3. Une inversion sur j_2 ($\mathcal{E}[\mathcal{B}] = \{j_2\}$)
4. Une double inversion sur j_1 et j_2 ($\mathcal{E}[\mathcal{B}] = \{j_1, j_2\}$)

Appliquer cette règle au décodage SSCF revient à limiter le nombre de calculs de métrique à un maximum de 3 pour chaque nœud de rendement 1. A noter qu'il est même possible de se limiter uniquement aux deux premières trajectoires – soit une seule métrique – sans que les performances du SSCF ne soient affectées significativement [2].

Le fait de diminuer le nombre de calculs de métrique permet également de réduire la complexité en réduisant le nombre d'inversions binaires susceptibles d'être intercalées dans la liste $\mathcal{L}_{\text{flip}}$. Par contre, il s'avère que cela ne rend pas l'identification de l'inversion binaire $\mathcal{E}_Y$ plus efficace dans la mesure où les inversions binaires écartées ont le plus souvent des métriques élevées.

Mentionnons que la réduction du nombre de métriques à calculer peut également s'obtenir sur la version non simplifiée du décodage à inversion. Dans [111], un critère simple est donné pour caractériser

[2] Une telle simplification ne devient préjudiciable que pour des taux d'erreurs que seul un décodeur par liste avec une taille de liste élevée est capable d'atteindre.

des inversions binaires dont la probabilité de corriger la trajectoire est négligeable, indépendamment de la réalisation de bruit. Cependant ce critère est lié aux blocs de rendement $R = 1$ et peut-être vu comme le pendant des optimisations du SSCF pour le décodage non-simplifié. Il consiste à dire que la première erreur ne survient que sur la première position d'un nœud de rendement $R = 1$. Si une inversion est pratiquée, alors la deuxième inversion est à rechercher sur la première position des nœuds de rendement $R = 1$ en considérant la position inversée comme un bit figé.

3.6 Autres considérations et perspectives

3.6.1 Sur le décodage à inversion

A la suite de notre travail sur le décodage à inversion, d'autres chercheurs se sont intéressés à cette stratégie de décodage, mais en empruntant une approche différente pour déterminer les tentatives.

Dans [32], un décodage à inversion partitionné est proposé, où le code est subdivisé en portions ayant chacune son propre CRC, et où un décodage à inversion est appliqué successivement sur chaque portion. Le décodeur utilisé est le SCFlip [3], ne pouvant donc pas corriger plus d'une erreur; mais le fait de subdiviser le code permet malgré tout, dans certains cas, de corriger plusieurs erreurs.

Une autre approche est basée sur l'utilisation de listes non ordonnées. La différence fondamentale avec les algorithmes SCFlip et D-SCFlip est qu'il n'y a plus de métrique ni une unique liste, mais une série de listes contenant les positions candidates pour la première erreur, la deuxième, etc... Le décodage consiste à tester les différentes combinaisons. Dans [31], les listes sont construites en instaurant un seuil sur les valeurs absolues des LLRs, la valeur du seuil étant obtenue par simulation Monte-Carlo, tandis que [111] propose un critère analytique pour discriminer les inversions binaires. L'avantage d'un telle approche est d'éviter d'avoir à trier les inversions binaires par métrique, son inconvénient étant que l'ordre des tentatives est sous-optimal et nécessite un nombre moyen de tentatives plus élevé que dans le décodeur proposé.

Pour réduire davantage la complexité du décodage à inversion, une idée pourrait être d'appliquer des bits de parités repartis le long du code, de manière à éviter que chaque tentative soit décodée jusqu'à la fin, et permettant d'écarter au plus tôt des trajectoires non pertinentes. A ce jour, une telle approche a été uniquement envisagée pour le décodage par liste dans [105] et a montré un gain substantiel par rapport au système concaténé CRC-polaire.

Le décodage simplifié proposé permet de réduire significativement la complexité et la latence de chaque tentative du décodage à inversion. Néanmoins, pour aller encore plus loin, la question de la gestion intelligente de la complexité d'une tentative à l'autre peut également être envisagée. En effet, il est clair que si chaque tentative recommence le décodage depuis les LLRs d'entrée du décodeur, un nombre significatifs de calculs de LLRs seront redondants. Dans [20], nous mentionnions qu'une nouvelle tentative pouvait recommencer le décodage uniquement après la première décision différent de celle de la tentative précédente, et ce sans avoir à augmenter la mémoire nécessaire. Une solution plus radicale pourrait être d'utiliser une pile pour stocker les résultats de tout ou partie des calculs effectués.

3.6.2 Décodage SCS vs D-SCFlip

Le décodage SCS nécessite également la comparaison de trajectoires de longueurs variables et il paraît dès lors légitime de s'interroger sur la pertinence de lui appliquer la métrique proposée pour le décodage à inversion. On considère un système concaténé Polaire-CRC avec un décodeur SCS de paramètres D, où D est la taille de la pile (le nombre maximal de trajectoires conservées en mémoire) [23]. Le décodage

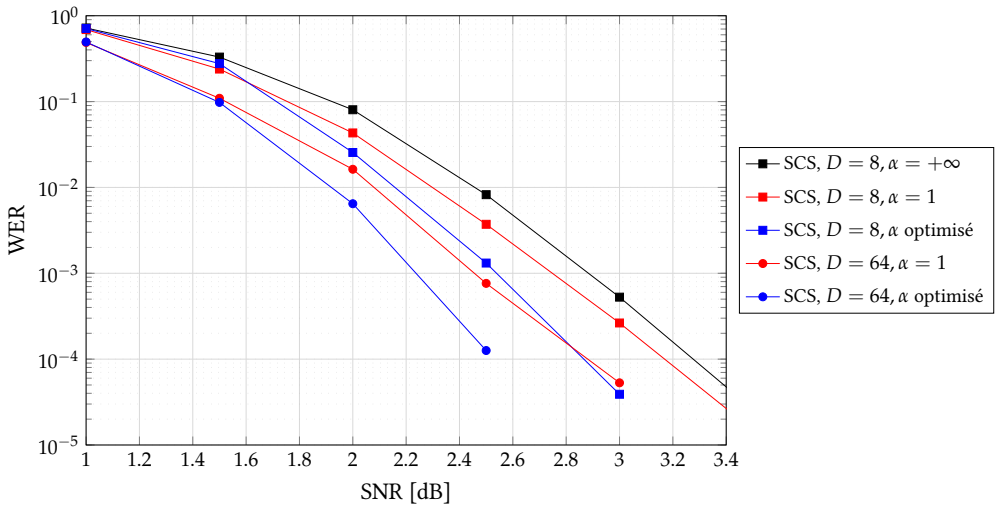


FIGURE 3.13: *Impact de la métrique du SCS pour un code* $(N = 1024, K = 512, r = 16)$

s'interrompt lorsque un chemin de longueur N vérifie de CRC où que D chemins de longueurs N ont été trouvés (avec ou sans vérification du CRC). L'impact de la métrique est donnée sur la figure 3.13. On observe un avantage substantiel à utiliser la métrique proposée (α optimisé par simulation), plutôt que celle du SCL ($\alpha = 1$) ou sa version simplifiée ($\alpha = +\infty$). Le paramètre le plus important est la valeur de D dans la mesure où la dégradation des performances est essentiellement due aux cas où le bon chemin n'est pas stocké dans la pile faute de place. L'utilisation de la métrique proposée permet de mieux conserver le bon chemin dans la pile et donc d'améliorer significativement les performances. Vu sous un autre angle, elle permet d'atteindre les mêmes performances pour une quantité de mémoire nettement réduite.

Dans [100], une comparaison de différentes métriques pour le SCS est fournie et montre des performances très similaires entre toutes les métriques envisagées, dans le cas où la taille de la mémoire est extrêmement élevée. Parmi ces métriques figure la version simplifiée du décodeur par liste (sans tenir compte des bits figés). Il s'avère que l'utilisation de cette métrique rend le décodeur très semblable à un décodeur à inversion pour une raison qu'on se propose d'expliciter maintenant.

Considérons le SCS utilisant la métrique simplifiée du SCL (sans tenir compte des bits figés). Si l'on considère la métrique associé à une chemin SC$\mathcal{E}$, correspondant à une inversion binaire $\mathcal{E} = \{i_1, \ldots, i_\omega\}$, alors on obtient:

$$M(\mathcal{E}) = \sum_{k=1}^{\omega} |L_{i_k}[\mathcal{E}]|, \tag{3.23}$$

et en particulier, la métrique de la trajectoire du SC est rigoureusement nulle. Rappelons que le décodeur séquentiel ne prolonge que la trajectoire ayant la plus faible métrique. Dès lors, il est clair que le décodeur considéré commencera directement par explorer la trajectoire du SC jusqu'à la fin. En cas d'erreur avéré, celui-ci se tournera vers la décision la moins fiable du SC (le LLR minimum en valeur absolue), puis prolongera cette trajectoire en se fiant aux signes des LLRs à nouveau jusqu'au bout, et ainsi de suite. On se rend compte finalement que ce décodeur a exactement la même stratégie d'exploration qu'un décodeur D-SCFlip avec $\alpha = +\infty$, mais où les trajectoires incomplètes sont stockées en mémoire afin d'éviter les redondances calculatoires. Si l'on considère maintenant la métrique exacte du SCL, toujours sans tenir compte des bits figés, ou la métrique proposée, il n'y a plus de rapprochement direct avec le décodeur D-SCFlip, mais les simulations tendent à montrer que les deux atteignent des performances proches lorsque $T = D$. L'avantage du SCS apparait pour des valeurs de D élevées, où il est en mesure de réduire le nombre de calculs de LLRs par rapport au D-SCFlip.

3.7 Conclusion

Ce chapitre s'est concentré sur le décodage à inversion des codes polaires, tant au niveau de l'optimisation des performances que de la réduction de complexité, à travers trois améliorations complémentaires. Tout d'abord, la limitation à une seule inversion générant une saturation préjudiciable des performances, un nouvel algorithme a été proposé, capable de corriger plusieurs erreurs par rapport au décodage SC. Celui-ci repose sur une métrique adaptée à l'exploration de trajectoires avec plusieurs inversions, et garantit d'explorer les celles-ci par probabilités de succès décroissantes. En second lieu, la métrique est optimisée de manière à mieux prendre en compte la nature séquentielle du décodeur par annulation successive, et offre une amélioration significative de l'identification des erreurs de décodage, réduisant aussi bien le taux d'erreur que le nombre moyen de tentatives. Finalement, sur la base des optimisations algorithmiques employées pour le décodage par liste, un décodeur à inversion simplifié a été proposé permettant la réduction significative des calculs de LLRs et de métriques pour une dégradation minime des performances.

Il est possible de faire un rapprochement entre le décodage SCS et le décodage à inversion. Il a été mis en évidence que la métrique proposée offre également un avantage pour ce décodeur en améliorant les performances et/ou permettant de réduire la mémoire nécessaire.

4 Codes Polaires pour les modulations d'ordre supérieur

Contenu du Chapitre

Ce chapitre propose d'explorer la question des modulations codées, combinant un code polaire binaire avec une modulation d'ordre supérieur. Notons que l'utilisation d'une modulation d'ordre supérieur implique que l'alphabet d'entrée du canal n'est plus binaire, contrairement aux chapitres précédents, et que le phénomène de polarisation d'un tel canal n'as pas été discuté dans cette thèse jusqu'ici. Néanmoins, l'utilisation d'un code polaire binaire permet de réduire le problème de polarisation (et donc de construction du code) à celui de plusieurs canaux à entrée binaire, correspondant à un ou à l'entrelacement de plusieurs niveaux binaires de la modulation, selon la technique de modulation codée considérée (multi-niveau ou à entrelacement de bits).

4.1 Introduction

La capacité d'un canal AWGN est donnée par:

$$C = \log_2(1 + \text{SNR}) \quad [bit/s/Hz] \tag{4.1}$$

Cette capacité décrit l'efficacité spectrale maximale pouvant être atteinte sur un canal AWGN en fonction du rapport signal à bruit (SNR, pour Signal-to-Noise Ratio, en anglais).

La démonstration de l'atteignabilité de la capacité est donnée par Shannon pour un l'ensemble des codes aléatoire, où la distribution du signal en entrée du canal est gaussienne. En pratique cependant, les systèmes utilisent une constellation constituée d'un ensemble de M points dans le plan complexe, chacun correspondant à un symbole à transmettre. A chacun de ces points de constellation est associé un "label", *i.e.* une séquence de $m = \log_2(M)$ bits. Une modulation codée utilisant un ou plusieurs codes binaires consiste à obtenir une séquence de mN bits codés, divisée en N bloc de m bits, et d'obtenir la liste des symboles de la constellation à transmettre.

Il existe différents types de constellation, dont la forme est liée à la manière dont le signal est modulé lors de la transmission : en fréquence (FSK, pour Frequency Shift Keying), en phase (PSK, pour Phase Shift Keying), en Amplitude (ASK, pour Amplitude Shift Keying) ou une combinaison de ces deux dernières, nommée QAM (pour Quadrature Amplitude Modulation en anglais). Cette dernière étant, de loin, la plus répandue en pratique.

Le fait de restreindre le nombre de bits par symbole induit une limitation de la capacité du canal à exactement $m = \log_2(M)$. De plus, on fera l'hypothèse tout au long de ce chapitre que les symboles de la constellation sont équiprobables. La figure 4.1 présente les valeurs des capacités symétriques[1] (en bit/symbole) pour les principales modulations.

Dans la littérature, on distingue en général trois grandes approches pour associer un code correcteur binaire avec une modulation m-aire, dénommées par ordre chronologique modulation en treillis (TCM, pour Treillis Coded Modulation, en anglais) [102], modulation multi-niveaux (MLCM, pour Multilevel Coded Modulation, en anglais) [47] et finalement modulation à entrelacement de bits (BICM, pour Bit-Interleaved Coded Modulation, en anglais) [18]. La TCM s'appliquant essentiellement en lien avec des codes convolutifs, il n'en sera pas question dans ce chapitre. Par contre, nous reviendrons sur les modulations BICM et MLCM, toutes deux considérées pour les codes polaires notamment dans [92] et [88] respectivement.

Ce chapitre reviendra sur le principe de la BICM et de la MLCM et leurs utilisations avec les codes polaires et s'attellera à mettre en évidence les avantages de l'utilisation des codes polaires pour la MLCM, où

[1] Contrairement au cas d'un canal à entrée binaire, la capacité symétrique n'est plus égale à la capacité du canal. Les solutions pour recouvrer cette perte sont mentionnées en perspectives de ce chapitre.

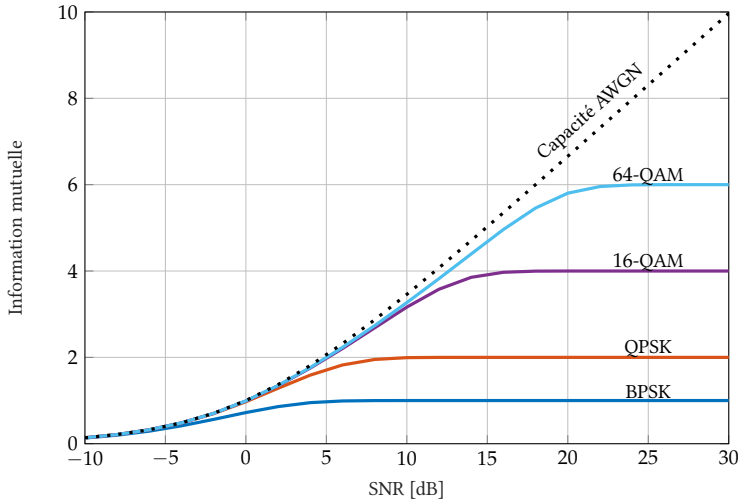


FIGURE 4.1: *Informations mutuelles des modulation M-QAM*

le problème du choix des rendements de codage par niveau binaire de la modulation[2] peut être facilement résolu par évolution de densité pour un décodeur SC. De plus, nous proposerons une méthode de construction basée sur la borne de l'union des codes, permettant d'optimiser les performances du décodeur SCL. Finalement, nous montrerons que la structure récursive des codes polaires peut se conjuguer efficacement avec la modulation MLCM, de manière à améliorer les performances de décodages des algorithmes basés sur le SC (décodage par liste, décodage à inversion, décodage séquentiel) tout en simplifiant considérablement le problème de l'optimisation des rendements de codage par niveau binaire de modulation.

4.2 Modulation multi-niveaux et modulation à entrelacement de bits

On notera par $\mathcal{C}$ la constellation complexe, de dimension $|\mathcal{C}| = M = 2^m$. Le paramètre M correspond au nombre de symboles complexes de la constellation, tandis que m, appelé l' *ordre* de la modulation, indique le nombre des bits par symbole. On supposera que tous les symboles sont équiprobables. On considère de plus une *fonction d'attribution*[3]:

$$\mathcal{L} : \{0,1\}^m \to \mathcal{C}, \tag{4.2}$$

associant à chaque séquence de m bits codés $(x_1,\ldots,x_m)$ un symbole complexe de la constellation $s = \mathcal{L}(x_1,\ldots,x_m)$. On notera par N le nombre total de symboles transmis.

4.2.1 Modulation multi-niveaux

Dans l'approche multi-niveaux [47], les bits ayant la même position dans les symboles transmis sont protégés par un même code correcteur binaire. Plus précisément, une approche multi-niveaux est caractérisée par m *niveaux binaires*, correspondant aux m positions des bits dans le symbole transmis, et un code correcteur $\mathcal{C}_\ell(N,k_\ell,\mathcal{I}^{(\ell)})$, où $\ell \in \{0,\ldots,m-1\}$ est associé à chaque niveau binaire. Le processus d'encodage d'une MLCM est présenté dans la figure 4.2. Les K bits d'information sont répartis sur les

[2]Par niveau binaire de modulation, nous entendons l'ensemble des bits ayant une même position dans les symboles transmis.
[3]Également appelé codage (e.g., codage de Gray), mais on évitera d'utiliser ce terme déjà utilisé dans un autre sens.

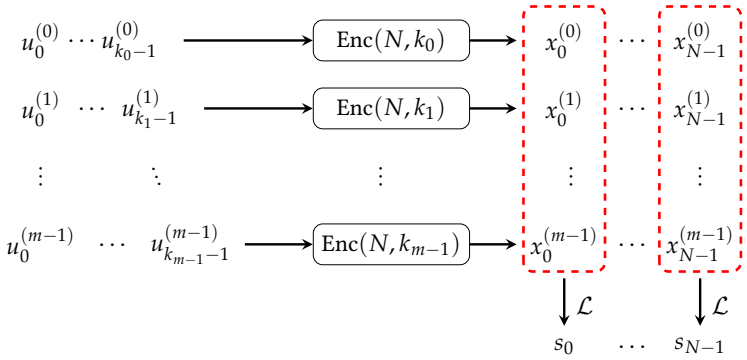


FIGURE 4.2: *Structure d'encodage et de modulation des bits d'information en MLCM*

m niveaux binaires, de rendements respectifs $k_0/N, k_1/N, \ldots k_{m-1}/N$ avec $K = \sum_{\ell=0}^{m-1} k_\ell$. Pour chaque niveau $\ell \in \{0,\ldots,m-1\}$, les bits d'information, notés $(u_0^{(\ell)},\ldots,u_{k_\ell-1}^{(\ell)})$, sont encodés pour obtenir les bits codés $(x_0^{(\ell)},\ldots,x_{N-1}^{(\ell)})$. Par suite, les symboles à transmettre sont obtenus en appliquant la fonction d'attribution $\mathcal{L}$ sur les bits codés $(x_j^{(0)},\ldots,x_j^{(m-1)})$, de sorte que chaque symbole contient un seul bit codé de chaque niveau binaire.

L'intérêt de la structure de la MLCM est de permettre l'application d'un *décodage multi-niveaux* à la réception, présenté sur la figure 4.4, et consistant à successivement démoduler puis décoder les niveaux $\ell \in \{0,\ldots,m-1\}$, en exploitant le résultat du décodage des précédents niveaux, d'après les étape suivantes (où $\bar{s}_i$ correspond au i-ème symbole reçu):

- Le tout premier niveau binaire $\ell = 0$ est démodulé puis décodé d'après:

$$\lambda(x_i^{(0)}) = \log\left(\frac{\Pr(x_i^{(0)} = 0 \mid \bar{s}_i)}{\Pr(x_i^{(0)} = 1 \mid \bar{s}_i)}\right) \quad \forall i \in \{0,\ldots,N-1\} \tag{4.3}$$

$$\hat{x}_{0:N-1}^{(0)} = \mathrm{Dec}(\lambda(x_0^{(0)}),\ldots,\lambda(x_{N-1}^{(0)})) \tag{4.4}$$

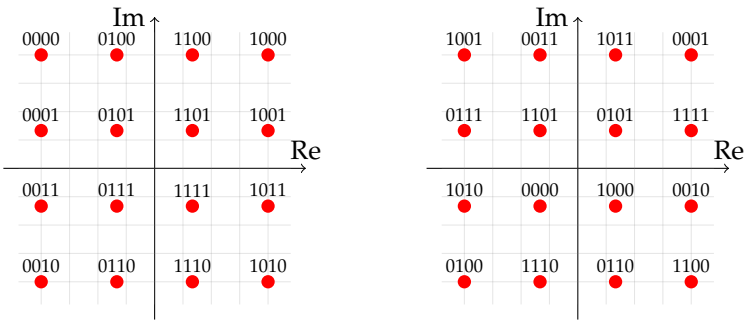
- Pour les niveaux binaires $\ell > 0$, la démodulation prend en compte les résultats des décodages des précédents niveaux:

$$\lambda(x_i^{(\ell)}) = \log\left(\frac{\Pr(x_i^{(\ell)} = 0 \mid \bar{s}_i, \hat{x}_{0:N-1}^{(0:\ell-1)})}{\Pr(x_i^{(\ell)} = 1 \mid \bar{s}_i, \hat{x}_{0:N-1}^{(0:\ell-1))}}\right) \quad \forall i \in \{0,\ldots,N-1\} \tag{4.5}$$

$$\hat{x}_{0:N-1}^{(\ell)} = \mathrm{Dec}(\lambda(x_0^{(\ell)}),\ldots,\lambda(x_{N-1}^{(\ell)})) \tag{4.6}$$

Raffinant les travaux précédents sur la MLCM [45], Wachsmann a montré dans [103, Th.1] que la MLCM, associée à un décodage multi-niveaux, atteint la capacité du schéma de modulation (indépendamment de la fonction d'attribution), si et seulement si le rendement de codage sur chaque niveau binaire est égal à la capacité du (canal à entrée binaire associé au) niveau. Une conséquence significative de ce théorème est que la capacité symétrique peut être atteinte sans l'utilisation de code non-binaire, mais simplement à partir d'un décodage multi-niveaux de codes binaires. Notons toutefois que si la fonction d'attribution n'a pas d'impact asymptotiquement (en termes de capacité), il s'avère que pour des codes de longueur finie, celle-ci joue un rôle essentiel. La solution offrant les meilleures performances en pratique est appelée SP (pour Set-Partitionning, en anglais) [102] et repose sur le principe de maximisation des distances euclidiennes entre les points de la constellation à chaque niveau binaire.

La figure 4.3(b) présente l'exemple du SP pour la constellation 16-QAM. La figure 4.5 illustre quant à elle les sous-constellations obtenues au fur et à mesure que le décodage avance d'un niveau binaire à un



(a) Attribution de Gray (b) Attribution SP

FIGURE 4.3: *Principales fonctions d'attribution pour une constellation* 16*-QAM*

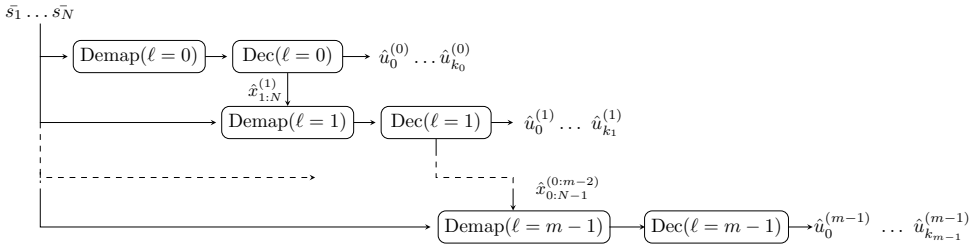


FIGURE 4.4: *Structure de décodage et de démodulation des bits d'information en MLCM*

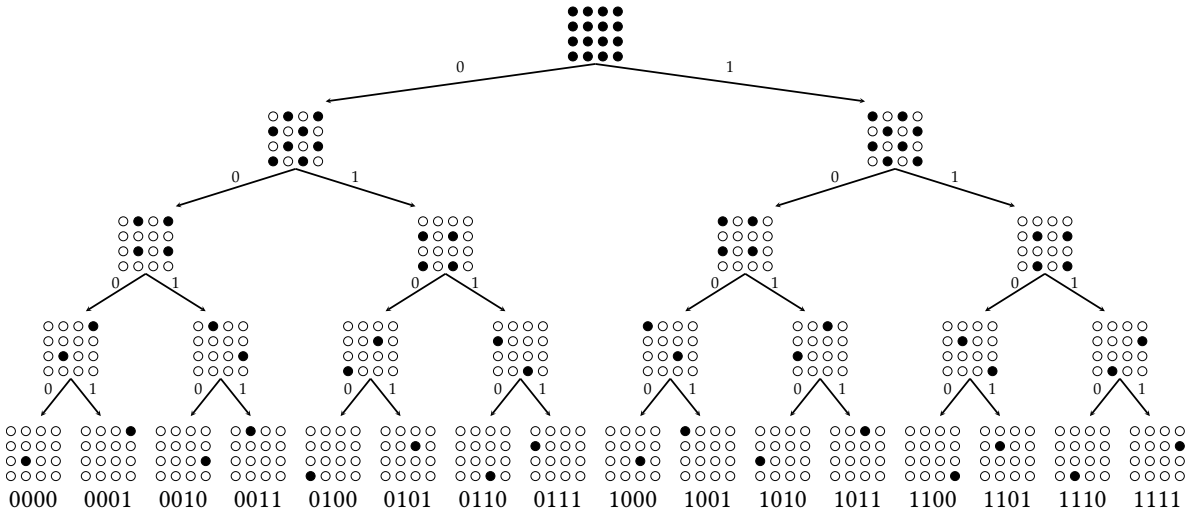


FIGURE 4.5: *Constellation SP dans une modulation multi-niveaux (*16*-QAM). En noir, les points de constellation pris en compte lors de la démodulation, en fonction de ce qui a été décodé sur les précédents niveaux. Il peut être observé que la distance minimale entre deux points augmente d'un facteur* $\sqrt{2}$ *à chaque niveau binaire.*

autre, pour une constellation 16-QAM. Puisque la distance euclidienne augmente avec le niveau binaire, il en sera de même pour la capacité des canaux correspondant aux différents niveaux. Il est attendu donc que la distribution des bits d'information sur les différent niveaux binaires vérifie $k_0 \leq k_1 \leq \cdots \leq k_{m-1}$, correspondant à des codes polaires de rendement croissant.

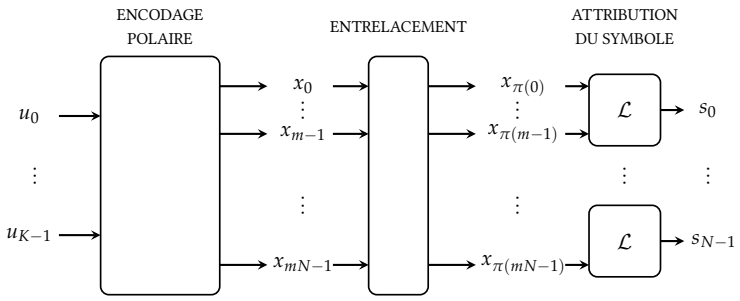


FIGURE 4.6: *Structure d'encodage et de modulation des bits d'information en BICM*

4.2.2 MODULATION À ENTRELACEMENT DE BITS

Une fonction de attribution couramment utilisée en pratique est la fonction d'attribution de Gray (appelée également codage de Gray), dû notamment aux avantages qu'elle procure pour la conception de circuits de conversion analogique-numérique. Une caractéristique de cette fonction d'attribution et que les séquences binaires correspondant à deux symboles voisins de la constellation complexe ne diffèrent que par un seul bit, maximisant ainsi le taux d'erreur binaire des systèmes non-codés. Un exemple de la fonction d'attribution de Gray pour la 16-QAM ($m = 4$) est donné dans la figure 4.3(a). A l'inverse du SP, où la distance euclidienne minimale augmente à chaque niveau binaire, celle-ci reste constante pour la fonction d'attribution de Gray. Cela se traduit par le fait que la connaissance des résultats de décodage des précédents niveaux n'apporte pas d'avantage significatif. Dès lors, il a été proposé de coder et démoduler tous les bits parallèlement, sans tenir compte du niveau binaire. Ainsi, un unique code de longueur mN est utilisé, et un entrelaceur est appliqué avant l'attribution des symboles à partir des bits codés. Le schéma d'encodage de cette technique, appelée BICM [18], est présentée dans la figure 4.6. Le récepteur est constitué d'un unique décodeur polaire de longueur $m \cdot N$. Les LLRs d'entrée sont obtenus par le dé-entrelacement des LLRs calculés par l'opération de démodulation:

$$\lambda(x_{\pi(i)}) = \log\left(\frac{\Pr(x_{\pi(i)} = 0 \mid \bar{s}_i)}{\Pr(x_{\pi(i)} = 1 \mid \bar{s}_i)}\right), \tag{4.7}$$

où $\bar{s}_i$ correspond au i-ième symbole reçu.

Du fait de la démodulation parallèle des bits, indépendamment du niveau binaire, la BICM induit une perte en terme de capacité par rapport à la MLCM. Cependant, il a été montré dans [18], que, à condition que la fonction d'attribution de Gray soit effectivement utilisé, cette perte reste très limitée. La figure 4.7 illustre cette très légère perte pour une modulation 16-QAM.

Notons tout de suite que la longueur du code est donnée par mN, de sorte que lorsque m n'est pas une puissance deux, l'utilisation des techniques du Chapitre 2 (poinçonnage, raccourcissement ou structures multi-noyaux) devient nécessaire pour les codes polaires. L'utilisation d'autres noyaux est envisagée dans [92] et la question du choix de l'entrelaceur est également discutée. Les questions du poinçonnage, ainsi que de l'optimisation de l'entrelaceur sont abordées dans [21]. La construction du code en BICM, *i.e.*, le choix des positions d'information, est obtenue en ré-utilisant les techniques de la Section 1.3.2 pour les canaux à entrée binaire. Si la méthode heuristique peut toujours être appliquée, les techniques par évolution de densité (DE) sont privilégiées dans la littérature [89, 48]. Précisons tout de suite que la méthode DE suppose que les LLRs en entrée du décodeur sont tous indépendants. Ce n'est cependant pas le cas en pratique, puisque m LLRs sont calculées à partir d'un même symbole reçu. Néanmoins, la pratique montre que le fait de ne pas tenir compte de ces dépendances n'empêche pas l'ED de fournir une solution efficace. L'approximation gaussienne s'est imposée comme la méthode de DE principale, du fait de sa faible

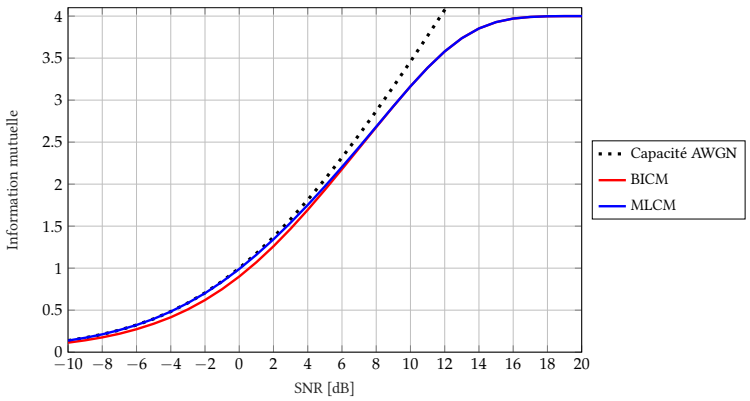


FIGURE 4.7: *Valeurs des capacités symétriques de la MLCM et BICM en* 16*-QAM*

complexité et sa simplicité de mise en œuvre. Elle consiste à reconstituer, par exemple de manière heuristique, les distributions des LLRs à l'entrée du décodeur, puis à les assimiler à des distributions gaussiennes, de manière à ce que les deux distributions (originale et assimilée) correspondent à deux canaux à entrée binaire de même capacité. En d'autre termes, le canal correspondant à chaque niveau binaire est approximé par un canal gaussien à entrée binaire, de même capacité. La construction du code s'obtient alors exactement comme indiqué dans la Section 1.3.2, pour un canal gaussien à entrée binaire.

4.2.3 COMPARAISON MLCM ET BICM

Du fait que la BICM induit une perte en termes de capacité, la MLCM offre de meilleure performances en général. Cette amélioration a été mise en évidence pour les codes polaires avec le décodeur SC, notamment dans [89], puis plus tard dans [97]. Qui plus est, la MLCM offre une réduction non négligeable en termes de complexité lorsque le SC est utilisé, puisque celle-ci est de l'ordre de $\mathcal{O}(mN\log(mN))$ pour la BICM, contre $\mathcal{O}(mN\log(N))$ pour la MLCM.

Pourtant, la stratégie BICM est devenue incontournable dans la grande majorité des standards actuels. Les principales raisons sont les suivantes:

1. Par rapport à la BICM, la MLCM a l'inconvénient de nécessiter l'utilisation de m rendements de codage distincts, un par niveau binaire, ce qui est délicat lorsque le code correcteur utilisé ne dispose pas d'une grande flexibilité.
2. Si asymptotiquement le choix optimal des m rendements de codage est donné par la capacité des m niveaux binaires correspondants, il s'avère qu'une optimisation plus fine – et capable d'optimiser conjointement les m rendements de codage – est souvent requise en longueur finie. Cependant, il n'existe pas toujours de méthode efficace d'optimisation en longueur finie, notamment lorsque des codes LDPC ou Turbo codes sont utilisés.
3. La capacité des niveaux binaires augmente significativement avec ℓ de sorte que les valeurs des rendements varient drastiquement, depuis un rendement très faible pour $\ell = 0$ à un rendement très élevé pour $\ell = m - 1$. Or, il est en général difficile de garantir un gain de codage notable sur une gamme si large de rendements

Les deux premiers points s'avèrent nettement moins contraignants lorsque les codes polaires sont considérés. Tout d'abord, ceux-ci disposent d'une excellente flexibilité en fonction du rendement. Deuxièmement, les performances en longueurs finie du SC peuvent être facilement estimées par évolution de densité, de sorte que, pour le décodeur SC, le choix des rendements ainsi que la construction des codes polaires n'est pas plus compliqué en MLCM qu'en BICM (cf. section suivante). Finalement, la question de

la variabilité des rendements n'est pas un problème pour le SC, grâce à une méthode efficace de construction. Cependant, cela impacte sérieusement le décodeur SCL, du fait que le gain reste très limité pour les rendement proches de 0 ou 1 du fait que la borne ML est très proche des performances du SC.

Précisons que le problème devient particulièrement délicat si l'on envisage l'utilisation de décodeurs CA-SCL sur chaque niveau binaire, du fait qu'il devient nécessaire de choisir, pour chaque niveau, le nombre optimal de bits de CRC (pour des rendements très faibles ou très élevés, un choix sous-optimal peut être excessivement dommageable), puis de s'efforcer de choisir les rendements des codes. Hormis utiliser une approche heuristique, une telle optimisation semble excessivement délicate.

Dans la partie suivante, nous reviendrons sur la méthode de construction des codes pour le décodeur SC et proposerons une nouvelle méthode pour le cas du décodeur SCL.

4.3 Construction du code polaire pour la MLCM

Dans [103], pas moins de cinq méthodes distinctes sont recensées pour résoudre le problème du choix des rendements des codes en MLCM, chacune utilisant un critère distinct. Parmi celles-ci, nous ne retiendrons que les trois suivantes (les deux autres s'appliquant plutôt à des codes aléatoires ou convolutifs):

1. Choisir les rendements égaux aux capacités des niveaux binaires correspondants
2. Équilibrer les valeurs des $(D^{\ell}_{\min})^2 \cdot \delta_{\ell}$, où $D^{\ell}_{\min}$ est la distance de hamming minimale du code du niveau binaire ℓ et δ_{ℓ} est la distance euclidienne minimale entre les points de la constellation (dans le cas du SP sur une constellation M-QAM, on a $\delta_{\ell+1}/\delta_{\ell} = \sqrt{2}$)
3. Équilibrer les taux d'erreurs sur chaque niveau binaire pour le décodeur considéré

C'est cette dernière règle qui est généralement utilisée pour faire la construction du code pour le décodeur SC. Notons toutefois qu'il a été montré dans [88] que celle-ci s'avère en fait équivalente au critère des capacités pour le décodage SC. Nous proposerons d'utiliser le même critère afin d'optimiser la construction pour un décodeur SCL.

4.3.1 Construction pour le décodeur SC

L'application de la 3-ième règle mentionnée ci-dessus requière de connaitre les performances – ou au moins une estimation précise – d'un code pour un rendement donné. Il s'avère que dans le cas des codes polaires avec un décodeur SC, une telle estimation peut être dérivée par simulation Monte-Carlo ou par évolution de densité, à partir des valeurs des $p_{i|}^{(\ell)}$, en utilisant la notation de la Section 1.3.2, où l'exposant (ℓ) est utilisé pour indiquer le niveau binaire correspondant. De plus, les similitudes entre les codes polaires et une structure multi-niveaux, mises en évidence dans [89], permettent de formaliser simplement le critère pour la construction des codes polaires sur les différents niveaux binaires, grâce à la borne supérieure suivante:

$$\mathrm{WER}_{\mathrm{SC}} \leq \sum_{\ell=0}^{m-1} \sum_{i \in \mathcal{I}^{(\ell)}} p_{i|}^{(\ell)} \tag{4.8}$$

Cette dernière formule ne présente pas seulement un critère pour choisir les valeurs des rendements, mais plus encore pour choisir, pour chaque niveaux binaires, les positions des bits d'information. En effet, tout comme pour un canal binaire où il s'agissait de trouver les K valeurs de $p_{i|}$ les plus faibles, les positions d'information sont choisies comme les K ayant les plus faibles valeurs de $p_{i|}^{\ell}$. La valeur de k_{ℓ} est ensuite déterminée en dénombrant les bits d'information sur chaque niveau binaire.

Pour calculer les $p_{i|}^{(\ell)}$, il est possible d'utiliser la méthode par simulation Monte-Carlo, mais la méthode privilégiée en pratique reste l'évolution de densité. Pour cela, la PDF des LLRs d'entrée (précisons que tous

les LLRs d'entrée d'un même niveau binaire suivent la même distribution) sont estimées par Monte-Carlo. Il est alors possible de propager cette PDF d'après les méthodes mentionnées dans le Chapitre 1. Il est également possible d'appliquer l'approximation gaussienne en assimilant le canal correspondant au ℓ-ième niveau binaire à un canal gaussien à entrée binaire de même capacité [48]. Par la suite, on dénotera par $\sigma^{(\ell)}$ l'écart type du bruit blanc additif du canal gaussien assimilé.

4.3.2 Construction pour le décodeur SCL

Nous proposons maintenant une méthode pour optimiser la construction pour le décodeur SCL. Supposons tout d'abord qu'un décodeur atteignant exactement la borne ML soit utilisé sur chaque niveau binaire. Dès lors, en notant $\mathrm{WER}_{ML}^{(\ell)}$ le taux d'erreur d'un tel décodeur sur le niveau binaire ℓ, en supposant les niveaux précédents corrects, le taux d'erreur total est donné par:

$$\mathrm{WER} = 1 - \prod_{\ell=0}^{n-1}(1 - \mathrm{WER}_{ML}^{(\ell)}) \approx \sum_{\ell=0}^{n-1} \mathrm{WER}_{ML}^{(\ell)} \tag{4.9}$$

Supposons qu'un code $\mathcal{C}^{(\ell)}(N,k_\ell,\mathcal{I}^{(\ell)})$ est utilisé sur un niveau binaire ℓ, alors la borne ML peut être estimée d'après la borne de l'union (cf. Section 1.2.3):

$$\overline{U_b}^{(\ell)} = \mu_{D_{\min}}^{(\ell)} Q\left(\frac{\sqrt{D_{\min}^{(\ell)}}}{\sigma^{(\ell)}}\right), \tag{4.10}$$

où $D_{\min}^{(\ell)}$ est la distance minimale de $\mathcal{C}^{(\ell)}$, $\mu_{D_{\min}}^{(\ell)}$ le nombre de mot de code de poids $D_{\min}^{(\ell)}$, et $\sigma^{(\ell)}$ est l'écart type du canal gaussien à entré binaire, ayant la même capacité que le canal correspondant au ℓ-ième niveau binaire.

En combinant les équations (4.9) et (4.10), il apparaît qu'un critère possible consiste à équilibrer les valeurs des $\overline{U_b}^{(\ell)}$. Le Chapitre 1 a fourni une méthode de faible complexité pour calculer $D_{\min}$ puis $\mu_{D_{\min}}^{(\ell)}$ pour un code polaire donné, de sorte que ce problème d'optimisation peut être efficacement résolu.

Il a été montré que les codes polaires ayant, pour tout couple (N,K), la plus grande distance minimale et le plus faible nombre de mots de code de poids faible sont donnés par les codes SDO définis dans le Chapitre 1. Dès lors, le problème d'optimisation revient à trouver les codes SDO équilibrant les valeurs des $\overline{U_b}^{(\ell)}$. Une solution efficace consiste à utiliser un algorithme itératif tel que présenté dans l'algorithme 7. Celui-ci est muni de la liste "ListeCV", contenant les indices des canaux virtuels $i \in \{0,\dots,N-1\}$, de manière à ce que le code obtenu en utilisant les $k < N$ premiers indices pour les bits d'information soit un code SDO. Une telle liste peut être obtenue d'après la métrique définie dans l'équation (1.41). On définit également la fonction UBcalc(ListeCV,k,σ) calculant, pour le code dont les bits d'information sont positionnés sur les k premiers indices de la liste ListeCV, la distance minimale, puis la multiplicité, puis finalement l'estimation de la borne de l'union par l'équation (4.10). A chaque itération, l'algorithme calcule $\overline{U_b}^{(\ell)}$ pour tout niveau binaire avec k_ℓ bits d'information, recherche les niveaux binaires ayant la valeur la plus élevée (ℓ_{max}) et la plus basse (ℓ_{min}) respectivement, puis décrémente $k_{\ell_{max}}$ et incrémente $k_{\ell_{min}}$. Ce processus est itéré tant que la somme $\sum_\ell \overline{U_b}^{(\ell)}$ continue de décroître.

La construction obtenue est optimisée pour un décodeur garantissant les performances ML. Si l'on considère le décodeur SCL des codes polaires (sans concaténation avec un CRC), cette propriété n'est pas vérifiée, mais l'on sait que l'on peut s'en approcher au fur et à mesure que la taille de la liste augmente. Mentionnons toutefois que dans le cas où le code utilisé s'avère très peu performant pour le décodeur SC, une taille de liste considérable risque d'être nécessaire. Pour trouver un compromis efficace, nous

Algorithme 7 Construction MLCM pour décodeur SCL

procédure CST-ML($\sigma_0,\ldots,\sigma_{m-1}$)
  Initialiser ListeCV;
  Initialiser $(k_0,k_1,\ldots,k_{m-1})$ tel que $\sum_\ell k_\ell = K$;
  Initialiser UB$= \sum_{\ell=0}^{m-1}$ UBcalc(ListeCV, k_ℓ,σ_ℓ);
  tant que (1) **faire**
    Trouver $\ell_{min} = \arg\min_{\ell\in\{0,\ldots,m-1\}}$ UBcalc(ListeCV, k_ℓ,σ_ℓ);
    Trouver $\ell_{max} = \arg\max_{\ell\in\{0,\ldots,m-1\}}$ UBcalc(ListeCV, k_ℓ,σ_ℓ)
    Calculer UB$_{new} = \sum_{\ell=0}^{m-1}$ UBcalc(ListeCV, $k_\ell + [\ell = \ell_{min}] - [\ell = \ell_{max}],\sigma_\ell$);
    si UB$_{new}$ < UB **alors**
      Faire $k_{\ell_{min}} = k_{\ell_{min}} + 1$;
      Faire $k_{\ell_{max}} = k_{\ell_{max}} - 1$;
      Faire UB = UB$_{new}$;
    sinon
      Sortir
    fin si
  fin tant que
  retourner $(k_0,k_1,\ldots,k_{m-1})$;
fin procédure

Remarque 1: La liste "ListeCV" est initialisée de manière à contenir la liste des indices $\{0,\ldots,N-1\}$, ordonnés d'après la métrique garantissant un code SDO, donnée dans l'équation (1.41) (pour optimiser la borne de l'union), ou d'après la métrique basée sur le développement β, d'après l'équation (1.40) (pour trouver un compromis entre performance SC et borne de l'union).
Remarque 2: L'initialisation des valeurs $(k_0,k_1,\ldots,k_{m-1})$ peut être faite d'après les valeurs obtenues pour le SC, de manière à limiter le nombre d'itérations.

proposons d'exploiter le rapprochement entre le critère de définition d'un code SDO et la méthode de construction utilisant le développement β, explicité dans la Section 1.3.2.3 du Chapitre 1. Rappelons que $\beta^{-1} = 0.84$ donne un code efficace pour le SC, tandis que l'on se rapproche d'un code SDO lorsque $\beta \to 1$. Pour construire un code polaire efficace pour le SCL, l'algorithme 7 n'est plus initialisé avec la liste "ListeCV" définie d'après le critère du code SDO, mais d'après le développement β pour une valeur $\beta \in [0.84,1[$. La valeur de β définissant un compromis efficace entre la performance SC (estimée par DE) et la performance ML (estimée d'après la formule (4.9)) peut s'obtenir facilement en testant différentes valeurs de β, dans la mesure où ces deux techniques sont de faible complexité. Notons que nous utilisons une seule et même liste ordonnée "ListeCV" pour tous les niveaux. Une granularité supplémentaire pourrait être obtenue en considérant une liste distincte pour chaque niveau binaire, mais les simulations montrent qu'un gain significatif est d'ores et déjà perceptible sans qu'il soit nécessaire de considérer un tel cas de figure.

L'impact des valeurs initiales des $(k_0,\ldots,k_{m-1})$ concerne essentiellement le nombre d'itérations de l'algorithme. En les choisissant égales aux valeurs obtenues pour le décodeur SC, le nombre d'itérations est réduit à une dizaine en général.

La figure 4.8 illustre l'avantage de la construction proposée sur une modulation 16-QAM avec $N = 256$ et $K = 512$ pour un canal AWGN. Alors que le SCL bute rapidement sur la borne ML lorsque la construction est optimisée pour le SC, la méthode proposée permet d'obtenir un gain de 0.5dB pour une taille de liste $L = 8$, tandis qu'un gain maximal de 0.7dB pourrait être obtenu lorsque des codes SDO

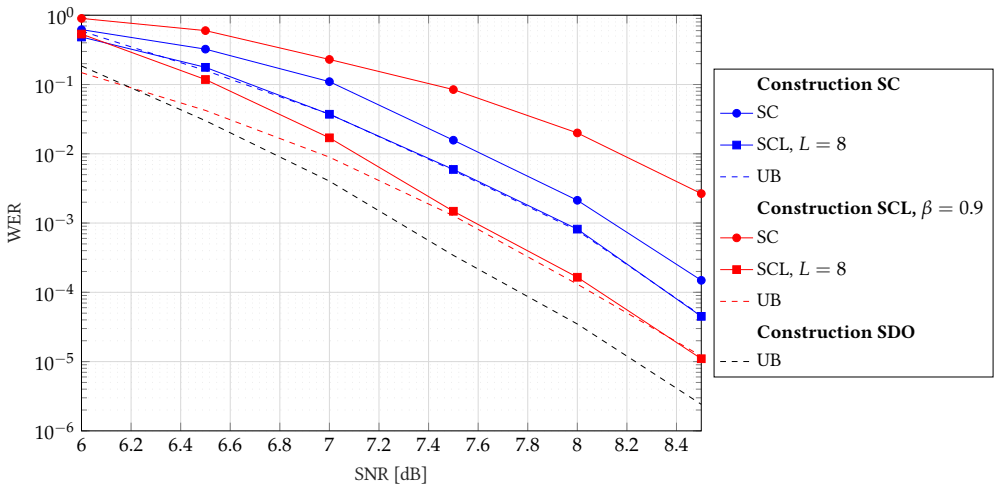


FIGURE 4.8: *Construction du code pour le SCL en MLCM pour une* 16*-QAM avec un canal AWGN. Le nombre de symboles est* $N = 256$ *et le rendement total vaut* $K/(mN) = 0.5$.

sont utilisés avec une taille de liste conséquente. Notons au passage que la borne de l'union utilisée pour réaliser l'optimisation des codes s'avère très précise, et conforte ainsi la méthode proposée.

4.4 Décodeurs multi-niveaux continus

Le parallèle entre la structure d'un code polaire est une structure multi-niveaux est bien connu [89], et nous proposons de l'exploiter dans la MLCM pour améliorer les performances de décodage des décodeurs polaires basés sur le SC (SCL, CA-SCL, D-SCFlip, SCS,...) au prix d'une augmentation légère de complexité. Contrairement au fonctionnement classique de la MLCM, où chaque niveau binaire dispose d'un décodeur spécifique et indépendant des autres niveaux binaires, la solution proposée consiste à considérer le décodage de tous les niveaux binaires de la MLCM à partir d'un unique décodeur, appelé décodeur continu. Dans le traitement classique de la MLCM, le décodage d'un niveau binaire fournit un unique candidat $(\hat{u}_0^{(\ell)}, \ldots, \hat{u}_{k_\ell}^{(\ell)})$, qui n'est par suite jamais remis en cause, de sorte qu'une erreur à un niveau binaire donné hypothèque les perspectives d'un décodage final correct. A l'inverse, les décodeurs continus considérés sont en mesure de revenir sur la décision d'un niveau binaire précédent de sorte qu'un gain significatif peut être obtenu.

L'approche proposée s'applique à tout décodeur dont le processus respecte l'aspect successif du SC. Nous détaillerons le cas du décodeur par liste (SCL et CA-SCL), ainsi que du décodeur D-SCFlip.

4.4.1 Décodage par liste continu

Considérons dans un premier temps un décodeur par liste générique, *i.e.* un décodeur retournant, à partir d'une unique séquence de LLRs d'entrée, un nombre L de candidats. De tels décodeurs existent pour presque toutes les familles de code, depuis les codes LDPC aux codes Turbo, en passant par les codes polaires. Pour améliorer les performances de la MLCM, il est possible de considérer un processus de décodage où les L candidats sont conservés à l'issue du décodage du premier niveau binaire, puis L démodulations sont effectuées en parallèle pour chacun des candidats, pour finalement appliquer L décodeurs par liste au niveau binaire suivant et ainsi de suite. Cette solution a l'inconvénient d'accroître considérablement la complexité, puisque celle-ci est multipliée par L à chaque niveau binaire.

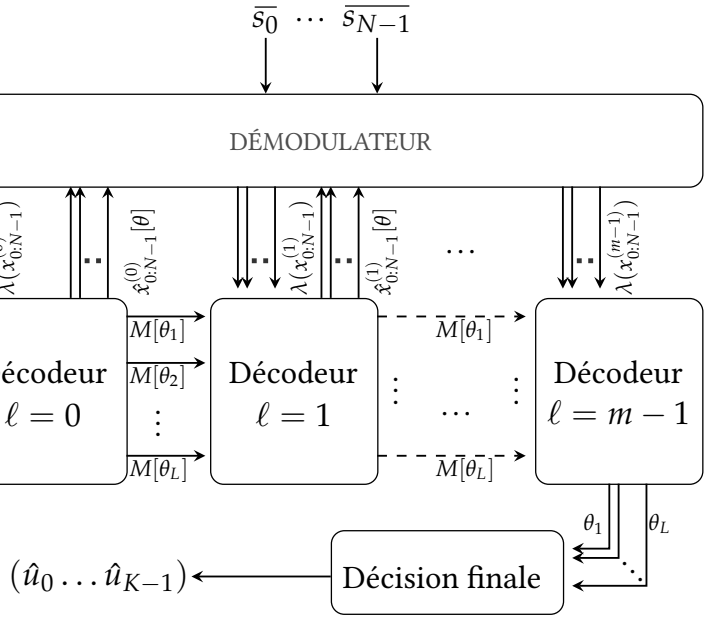


FIGURE 4.9: *Structure du décodeur SCL continu en MLCM*

Cependant, le décodeur par liste des codes polaires permet d'appliquer une approche similaire de manière significativement plus simple et pour une augmentation limitée de la complexité. La raison fondamentale est que ce décodeur peut s'initialiser avec plusieurs candidats, selon les étapes suivantes:

1. Démodulation du niveau binaire ℓ : démoduler parallèlement les L résultats de décodage du niveau précédant
2. Décodage du premier bit : calculer le LLR du premier bit d'information du niveau binaire ℓ, $\hat{u}_i^\ell$, à partir des L séquences d'entrée du décodeur
3. Calculer les métriques des $2L$ chemins et ne conserver que les L ayant la plus forte probabilité. Pour renforcer le processus de sélection des trajectoires les plus probables, la métrique d'une trajectoire au niveau binaire ℓ est initialisée à la valeur de la métrique à l'issue du niveau binaire $\ell - 1$.
4. Procéder de manière identiques pour les bits d'information suivants

La figure 4.9 présente la structure du décodeur SCL, et tout particulièrement l'articulation des décodeurs des différents niveaux binaires. Le décodeur d'un niveau ℓ reçoit du démodulateur les LLRs d'entrée $\lambda(x^{(\ell)}_{0:N-1})[\theta_m]$, calculés à l'issue de la démodulation lorsque les bits précédents correspondent à la trajectoire θ_m avec $m \in \{1, \dots, L\}$, présente dans la liste du décodeur du niveau binaire $\ell - 1$ à la toute fin de son décodage. Il reçoit également les valeurs des métriques obtenues lors du décodage du niveau binaire $\ell - 1$ pour chaque trajectoire θ_m. Muni de ces deux informations, le décodage du niveau ℓ est effectué, puis le même processus est itéré avec le niveau binaire suivant. La décision finale de la trajectoire, *i.e.* du mot de code $(\hat{u}_0, \dots, \hat{u}_{K-1})$ n'est opérée qu'à l'ultime fin du décodage de tous les niveaux binaires, soit à partir de la valeur de la métrique, soit au moyen d'un CRC.

Durant ce processus, il est très probable que les candidats en erreur à la fin du niveau binaire précédent soient écartés lors du décodage du niveau suivant, car trop peu probables. Ainsi, la décision du candidat à l'issue d'un niveau n'est pas effectuée immédiatement, mais durant le décodage du niveau suivant. Cela permet de ne pas utiliser de CRC sur chaque niveau binaire, dont l'impact pour les rendements faibles ou élevés peut être problématique. En revanche, rien n'empêche d'utiliser un unique CRC, servant à l'identification de la bonne trajectoire à l'issue du tout dernier niveau.

Comparativement au cas des décodeurs indépendants, la complexité du décodage continu augmente pour les raisons suivantes:

- le nombre de démodulations est multiplié par L pour les niveaux binaires $\ell > 0$.
- le décodage du premier bit des niveaux binaires $\ell > 0$ requière $L \cdot (N-1)$ calculs de LLRs, contre $N-1$ pour des décodeurs indépendants
- le décodage des $\lfloor \log_2(L) \rfloor - 1$ bits suivants requière un nombre de calculs de LLRs légèrement plus élevé, par rapport au cas où le nombre de trajectoires considérées par le décodeur SCL est encore strictement inférieur à L

Ainsi, la complexité augmente d'autant plus que la taille de la liste est élevée. Néanmoins, nous précisons que le nombre de démodulations nécessaires peut être réduit. En effet, pour un symbole reçu $\bar{s}_i$ avec $i \in \{0, \dots, N-1\}$, correspondant au symbole transmis $s_i = x_i^{(0:m-1)}$, la démodulation du niveau ℓ dépend uniquement de la valeur des bits codés $\hat{x}_i^{(0:\ell-1)}$ obtenus à l'issue du décodage de chacun des niveaux précédents. Dès lors, deux trajectoires θ_1 et θ_2 nécessitent des démodulations distinctes seulement lorsque $\hat{x}_i^{(0:\ell-1)}[\theta_1] \neq \hat{x}_i^{(0:\ell-1)}[\theta_2]$. Plus généralement, pour chaque symbole d'un niveau ℓ, un maximum de $2^{\ell-1}$ démodulations sont suffisantes pour obtenir les LLRs correspondants. Cet avantage est particulièrement intéressant pour une taille de liste élevée, où le nombre d'opérations de démodulation est significativement réduit.

La question de la construction du code pour un tel décodeur paraît nettement moins évidente que pour des décodeurs indépendants. Néanmoins, pour le CA-SCL les simulations montrent que la construction du code obtenue pour le SC pour $K + r$ bits, où r est le nombre de bits de CRC, s'avère très proche de l'optimal.

4.4.2 Décodage à inversion continu

Le principe d'un décodeur à inversion continu est encore plus simple que pour le décodeur par liste. Le décodeur D-SCFlip continu est muni d'une liste d'inversions binaires, correspondant à une série d'inversions pouvant porter sur différents niveaux binaires. Ainsi, il est en mesure de corriger des erreurs sur tout niveau binaire. La principale question relative à ce décodeur est de savoir si la métrique utilisée en canal gaussien à entrée binaire (Eq. (3.13) , Section 3.3) s'avère tout aussi convaincante pour la MLCM, et notamment, concernant le choix du paramètre α. Les simulations montrent cependant qu'un gain significatif de performance peut être obtenu sans qu'une ré-optimisation délicate ne soit nécessaire.

4.4.3 Résultats des simulations

La figure 4.10 compare les performances de de l'approche continue par rapport à l'utilisation de décodeurs indépendants sur chaque niveau binaire en MLCM, avec un décodeur CA-SCL. Le canal est AWGN, la constellation utilisée est une 16-QAM et le nombre de symboles codés est fixé à $N = 256$. Pour optimiser l'approche utilisant des décodeurs séparés, une optimisation heuristique est utilisée afin d'équilibrer les taux d'erreur des décodeurs sur chaque niveau binaire. Un gain substantiel est observé pour l'approche continue, et a tendance à augmenter avec la taille de la liste, pour atteindre près de 0.35dB pour $L = 8$ pour un taux d'erreur de 10^{-4}.

La figure 4.11 compare les modulations MLCM et BICM pour une constellation 16-QAM avec un canal AWGN et $N = 256$ symboles. Trois décodeurs sont considérés : le décodeur SC, le décodeur CA-SCL, et le décodeur D-SCFlip. Pour la MLCM, on considère uniquement l'approche continue. Les résultats mettent en évidence une supériorité significative de la MLCM, avec un gain de plus de 0.6dB pour chacun des décodeurs considérés pour un taux d'erreur de 10^{-3}.

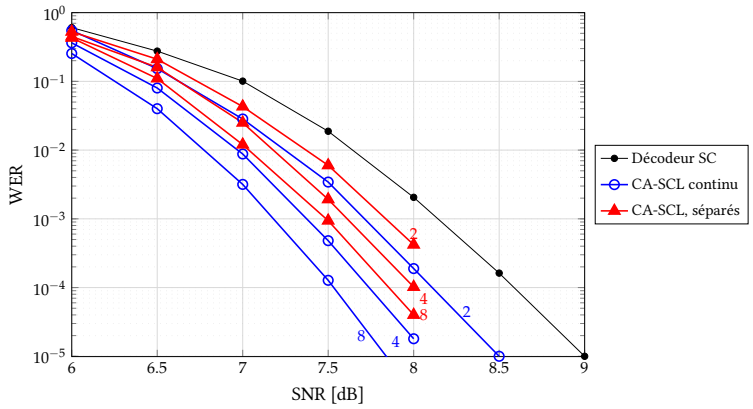


FIGURE 4.10: *Comparaison du décodeur CA-SCL continu avec des décodeurs séparés pour $N = 256$ sur une 16-QAM et un canal AWGN*

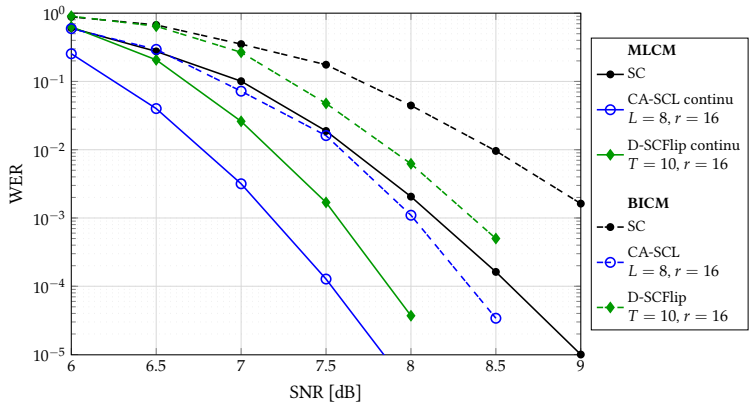


FIGURE 4.11: *Comparaisons de la BICM et de la MLCM sur une 16-QAM et un canal AWGN, avec $N = 256$*

4.5 CONCLUSION

Ce chapitre est revenu sur l'utilisation des codes polaires dans les modulations codées de type MLCM et BICM. Il a été mis en évidence que les codes polaires présentent des aptitudes uniques, rendant l'utilisation de la MLCM tout à fait attrayante. En effet, les principales limitations de la MLCM expliquant la suprématie de la BICM, sont essentiellement liées à la difficulté de choisir les valeurs des rendements des codes, ainsi que la variété des rendements obtenus. Les codes polaires, en plus d'être parfaitement flexible en fonction du rendement, disposent de deux méthodes d'analyses puissantes, que sont l'évolution de densité, et l'analyse des propriétés du spectre des codes pour obtenir une estimation précise de la borne ML. Nous avons montré comment exploiter ces deux techniques de manière à optimiser la construction du code pour le décodeur SCL.

Dans un second temps, le rapprochement entre la structure d'un code polaire et le concept de multi-niveaux est exploité afin d'élaborer une approche continue pour les décodeurs polaires basés sur le SC. Cette approche a été illustrée pour le décodage par liste des codes polaires, ainsi que le décodage D-SCFlip, et apporte un gain tangible en termes de performance.

Durant ce chapitre, les symboles étaient supposés uniformément distribués. Il est cependant connu qu'une telle hypothèse induit une perte par rapport à la capacité du canal. Une solution pour compenser cette perte, s'appliquant aussi bien en MLCM qu'en BICM, consiste à utiliser du *probabilistic shaping*. Il s'agit de modeler la probabilité des symboles de manière à se rapprocher de la distribution maximisant l'information mutuelle du canal. Cette technique a été appliquée au codes polaires en MLCM dans [77]

en BICM dans [49, 50] avec un gain substantiel à chaque fois.

5 Codes Polaires Non-binaires

Contenu du Chapitre

Ce chapitre étudie l'utilisation de codes polaires non-binaires, aussi bien pour un canal à entrée binaire que non-binaire. En particulier, les codes polaires binaires utilisant le noyau d'Arikan sont généralisés sur un alphabet fini $(\mathcal{A},+)$ muni d'une structure de groupe additif, à partir de la transformation $(u_0,u_1) \rightarrow (u_0+u_1,\pi(u_1))$, où $\pi : A \rightarrow A$ est une permutation. Un code non-binaire de longueur $N = 2^n, n \in \mathbb{N}^*$ est obtenu en appliquant cette transformation de manière récursive, et en utilisant des permutations potentiellement différentes à chaque étape de la récursion. Les algorithmes de décodage du cas binaires peuvent se généraliser naturellement aux code polaires non-binaires ainsi construits. En particulier, ce chapitre détaille une méthode pour optimiser les permutations utilisées lors de la construction récursive du code.

5.1 Polarisation d'un canal non-binaire

Soit W un canal non-binaire, discret, symétrique et sans mémoire, d'alphabet d'entrée $\mathcal{X}$, de cardinal $|\mathcal{X}| = M$, et d'alphabet de sortie $\mathcal{Y}$. On définit l'information mutuelle du canal W par la quantité:

$$I(W) \triangleq \sum_{x\in\mathcal{A}} \sum_{y\in\mathcal{Y}} \frac{1}{M} \log_M \left(\frac{W(y|x)}{\sum_{x'\in\mathcal{A}} \frac{1}{M} W(y|x')} \right) \tag{5.1}$$

On considérera par la suite que $\mathcal{X}$ est muni d'une loi additive +, telle que $(\mathcal{X},+)$ forme un groupe commutatif. Dans [85], le phénomène de polarisation est étudiée pour la transformation:

$$\begin{cases} x_0 &= u_0 + u_1 \\ x_1 &= u_1 \end{cases} \tag{5.2}$$

Il est démontré que, pour tout q premier, cette transformation *polarise* lorsqu'elle est utilisée de manière récursive, dans le sens où le processus stochastique $I_n = \{I(W_N^{(i)}), \forall i \in \{0,\dots,N-1\}, N = 2^n\}$, où $I(W_N^{(i)})$ est l'information mutuelle du i-ième canal virtuel, est une martingale et converge vers la variable aléatoire I_∞ telle que $\Pr(I_\infty = 1) = I(W)$ et $\Pr(I_\infty = 0) = 1 - I(W)$.

En revanche, ce résultat ne se généralise pas pour toute valeur de M. Une première solution proposée dans [85] est de décomposer W en une succession de canaux dont le cardinal est un nombre premier, similairement à un décodage multi-niveaux comme détaillé dans le Chapitre 4 (où un canal de dimension $M = 2^m$ était décomposé en m canaux de dimension 2). L'autre solution envisagée dans [85] consiste à considérer la transformation:

$$\begin{cases} x_0 &= u_0 + u_1 \\ x_1 &= \pi(u_1) \end{cases} \tag{5.3}$$

où $\pi : \mathcal{X} \rightarrow \mathcal{X}$ est une permutation. Cette transformation, que l'on qualifiera de *noyau*, est appliquée de manière récursive pour définir un code polaire non-binaire de longueur $N = 2^n$. Il est démontré qu'en sélectionnant la permutation π de manière aléatoire et uniformément parmi l'ensemble des permutations de $\mathcal{X}$ à chaque utilisation (dans la construction récursive), le processus obtenu polarise pour tout canal discret sans mémoire W.

Le cas des *noyaux* de dimension $\ell > 2$ est investigué dans [68], où une transformation linéaire $g : \mathcal{X}^\ell \rightarrow \mathcal{X}^\ell$ est envisagée avec un alphabet d'entrée $\mathcal{X}$ muni d'une structure d'anneau commutatif $(\mathcal{X},+,\cdot)$. Cela autorise à représenter la transformation récursive par une matrice carrée G, telle que $g(u_0^{\ell-1}) = u_0^{\ell-1} \cdot G$. Il est démontré que si le cardinal de $\mathcal{X}$ est premier, alors la transformation polarise. Lorsque le

cardinal de $\mathcal{X}$ n'est pas premier, une condition suffisante pour que la transformation polarise est que G ne soit pas diagonale et qu'il existe $k \in \{0,\dots,\ell-1\}$ et $j \in \{0,\dots,k-1\}$ tels que $G(k,j)$ soit un élément primitif (*i.e.* générateur du groupe multiplicatif).

Capitalisant sur ce résultat, [68] a proposé d'utiliser des matrices de code Reed-Solomon $G_{RS}(\ell)$. En généralisant la notion d'exposant d'erreur pour les canaux non-binaires, il est démontré celui des matrices $G_{RS}(\ell)$ excède facilement ceux des meilleures matrices binaires connues de longueurs $\ell < 32$, et tend vers 1 lorsque $\ell \to \infty$. Les performances de ces noyaux sont évaluées dans [69] et témoignent d'un gain substantiel par rapport aux codes binaires, pour un canal BI-AWGN. D'autres méthodes pour obtenir des noyaux avec un bon exposant d'erreur sont envisagées à partir des codes Reed-Muller généralisés ou des codes Hermitiens.

Si les noyaux obtenus peuvent être excellents en termes de performance, le problème de la complexité de décodage est un sérieux frein en pratique. Non seulement, un décodeur non-binaire requière un nombre de calculs significativement plus élevé qu'un décodeur binaire (cf section suivante), mais en plus, l'utilisation de noyaux de dimension $\ell > 2$ pose la même difficulté que pour le cas binaire, à savoir la gestion efficace des calculs intermédiaires, sans laquelle la complexité du décodeur est environ ℓ^2 fois supérieure. Une méthode de décodage a été proposée dans [99] à partir d'un décodeur de Stack, mais la complexité reste supérieure à celle des codes binaires d'un facteur 15 pour des performances comparables.

L'approche pragmatique consiste à se contenter de noyaux non-binaires de dimension $\ell = 2$, pour lesquels la complexité reste parfaitement abordable pour des valeurs de M limitées. Cette approche est adoptée dans la suite de ce chapitre.

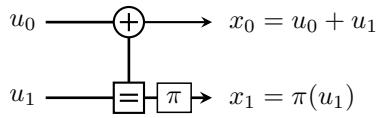
5.2 Codes polaires non-binaires

5.2.1 Définition d'un code non-binaire

Dans la suite on distinguera entre l'alphabet du code non-binaire, notée $\mathcal{A}$, et l'alphabet du canal, noté $\mathcal{X}$. Cela nous permettra en particulier de considérer aussi bien le cas où le code non-binaire est transmis sur un canal binaire (donc $\mathcal{X} = \mathbb{F}_2$), que celui où il est transmis sur un canal non-binaire (dans quel cas on pourra potentiellement avoir $\mathcal{X} = \mathcal{A}$).

Pour des raisons pratiques évidentes, on supposera que le cardinal de $\mathcal{A}$ est une puissance de 2, $|\mathcal{A}| = 2^p$, et sans perte de généralité on pourra choisir $\mathcal{A} = \mathbb{F}_2^p$. Un élément de $\mathcal{A}$, correspondant donc à une séquence de p bits, sera également appelé *symbole*. On note que $\mathcal{A}$ est muni d'une structure de groupe additif $(\mathcal{A},+)$, où l'addition $a+b$, avec $(a,b) \in \mathcal{A}^2$, correspond à l'opération du "ou exclusif" appliquée sur chaque bit des symboles a et b. On considère la transformation :

$$\begin{cases} x_0 &= u_0 + u_1 \\ x_1 &= \pi(u_1) \end{cases}$$



où $\pi \in \mathrm{Sym}(\mathcal{A})$ appartient au *groupe symétrique* de $\mathcal{A}$, *i.e.* le groupe des permutations de $\mathcal{A}$. Cette transformation sera appelée *noyau* du code polaire, dont le graphe est représenté ci-dessus.

Nous considérerons ici le cas le plus général d'un noyau défini par une permutation quelconque $\pi \in \mathrm{Sym}(\mathcal{A})$; on dira dans ce cas que le noyau est défini sur $\mathrm{Sym}(\mathcal{A})$. Néanmoins, il est possible de définir des ensembles plus restrictifs, en imposant des contraintes spécifiques sur la permutation $\pi \in \mathrm{Sym}(\mathcal{A})$. En rappelant que $\mathcal{A} = \mathbb{F}_2^p$, mais en privilégiant temporairement la notation $\mathbb{F}_2^p$ pour plus de clarté, on obtient les deux cas particuliers suivants:

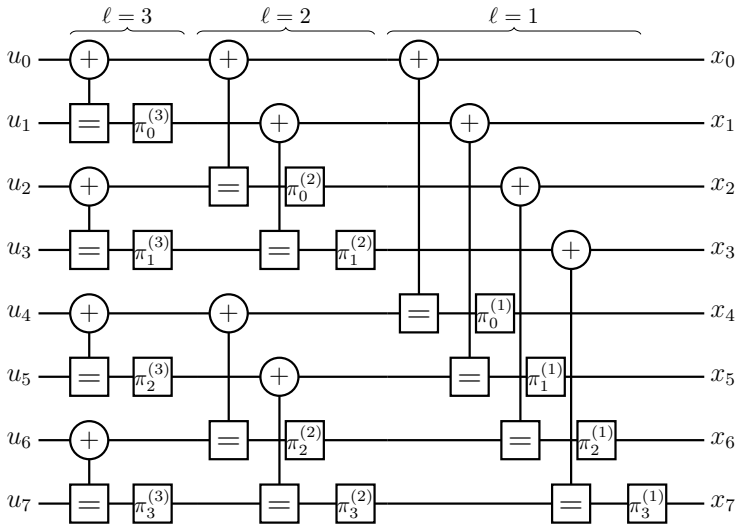


FIGURE 5.1: *Graphe d'un code non-binaire de longueur* $N = 8$

- Soit $\mathrm{GL}(\mathbb{F}_2, p)$ le groupe général linéaire d'ordre p, *i.e.*, le groupe multiplicatif des matrices inversible de taille $p \times p$, à coefficients dans $\mathbb{F}_2$. S'il existe $h \in \mathrm{GL}(\mathbb{F}_2, p)$ tel que $\pi(a) = h \cdot a, \forall a \in \mathbb{F}_2^p$, alors on écrira $\pi \in \mathrm{GL}(\mathbb{F}_2, p)$ et on dira que le noyau est défini sur $\mathrm{GL}(\mathbb{F}_2, p)$.
- Soit $\mathbb{F}_{2^p}$ le corps de Galois à 2^p éléments. Alors $\mathbb{F}_{2^p} = \mathbb{F}_2[X]/P(X)$ est le quotient de l'anneau des polynômes $\mathbb{F}_2[X]$ par l'idéal engendré par un polynôme irréductible $P(X)$ de degré p. Ainsi, chaque élément de $\mathbb{F}_{2^p}$ peut être représenté sous la forme d'un polynôme de degré strictement inférieur à p, et il existe un isomorphisme $(\mathbb{F}_{2^p}, +) \simeq (\mathbb{F}_2^p, +)$ identifiant un polynôme de degré strictement inférieur à p au vecteur de ses p coefficients. Une permutation $\pi \in \mathrm{Sym}(\mathbb{F}_2^p)$ s'identifie, via cet isomorphisme, à une permutation de $\mathbb{F}_{2^p}$, que nous allons continuer de noter π. S'il existe $h \in \mathbb{F}_{2^p}$ tel que $\pi(a) = h \cdot a, \forall a \in \mathbb{F}_{2^p}$, alors on écrira $\pi \in \mathbb{F}_{2^p}$ et on dira que le noyau est défini sur $\mathbb{F}_{2^p}$ (ici $h \cdot a$ dénote la multiplication dans le corps de Galois).

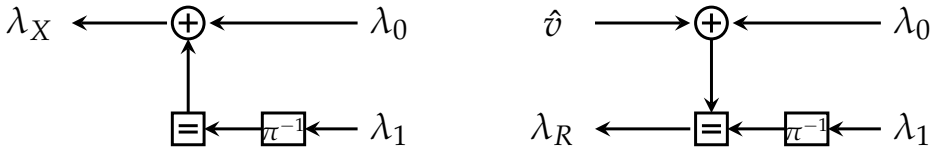
En réactualisant la notation $\mathcal{A} = \mathbb{F}_2^p$, nous noterons par la suite $\mathrm{GL}(\mathcal{A}) \triangleq \mathrm{GL}(\mathbb{F}_2, p)$ et $\mathrm{GF}(\mathcal{A}) \triangleq \mathbb{F}_{2^p}$.

Un code non-binaire de longueur $N = 2^n$ est obtenu en utilisant la transformation noyau de manière récursive, similairement au cas binaire. Un exemple du graphe de l'encodage non-binaire de longueur $N = 8$ est présenté sur la figure 5.1. Tout comme pour un code binaire, le décodage consiste à propager les messages (cf section suivante) depuis la droite de ce même graphe vers la gauche. Outre les nœuds xor et répétitions, chaque noyau du graphe contient un nœud de permutation, correspondant à l'application d'une permutation $\pi_i^{(\ell)} \in \mathrm{Sym}(\mathcal{A})$, possiblement distincte d'un noyau à un autre. Si tous les noyaux sont définis sur $\mathrm{GL}(\mathcal{A})$ (resp. $\mathrm{GF}(\mathcal{A})$) on dira que le code polaire est défini sur $\mathrm{GL}(\mathcal{A})$ (resp. $\mathrm{GF}(\mathcal{A})$).

On notera $(u_0, u_1, \ldots, u_{N-1}) \in \mathcal{A}^N$ les symboles d'entrée de l'encodeur, correspondant soit à des symboles figés, soit à des symboles d'information (voir aussi la section 5.2.3 concernant la construction du code). Les symboles codés seront notés par $(x_0, x_1, \ldots, x_{N-1}) \in \mathcal{A}^N$.

5.2.2 DÉCODAGE D'UN NOYAU NON-BINAIRE

Le décodage d'un code polaire non-binaire suit un processus très similaire au cas des codes binaires, consistant à décoder les symboles par indices croissants, en ré-utilisant les décisions précédentes. La différence repose sur le fait qu'il ne s'agit plus de propager dans le graphe des messages sous la forme d'un unique LLR, mais sous la forme d'un vecteur de $q \triangleq |\mathcal{A}|$ probabilités, noté $\lambda = \{\Pr(u = a \mid y), \forall a \in \mathcal{A}\}$,



(a) Nœud xor (b) Nœud répétition

FIGURE 5.2: *Règles de propagation des messages pour les nœuds xor (a) et répétition (b)*

chacune correspondant à la probabilité que la valeur du symbole $u \in \mathcal{A}$ soit égale à $a \in \mathcal{A}$. La propagation des messages obéit à deux fonctions de mise à jour, relatives aux nœuds xor et répétition respectivement. Considérons ces deux nœuds, présentés sur la figure 5.2, où λ_0 et λ_1 correspondent aux vecteurs de probabilités associés aux deux entrées du noyau, λ_X et λ_R aux messages de sortie des deux nœuds, et v la décision dure, issue du décodage des précédents symboles. La permutation π se traduit par une simple permutation du vecteur λ_1, à savoir $\lambda'_1(a) = \lambda_1(\pi(a))$, où $\lambda'_1(a)$ est la probabilité du symbole $a \in \mathcal{A}$ à la sortie du nœud de permutation. Le message λ_X en sortie du nœud xor s'obtient d'après:

$$\lambda_X(a) = \mu_X \cdot \sum_{b \in \mathcal{A}} \lambda_0(a+b) \cdot \lambda_1(\pi(b)), \quad \forall b \in \mathcal{A}, \tag{5.4}$$

où μ_X est un facteur de normalisation tel que $\sum_{a \in \mathcal{A}} \lambda_X(a) = 1$. A noter que, tout comme pour le cas binaire, il existe une approximation "Min-Sum" pour cette fonction de mise à jour [106]. L'implémentation directe de ce calcul est de complexité q^2, mais il a été montré que celle-ci peut se réduire à $q\log(q)$ en utilisant la transformée de Walsch-Hadamard [28].

Pour le calcul du message λ_R, en sortie du nœud répétition, la fonction de mise à jour est donnée par:

$$\lambda_R(a) = \mu_R \cdot \lambda_0(a+\hat{v}) \cdot \lambda_1(\pi(a)), \tag{5.5}$$

où μ_R est un facteur de normalisation tel que $\sum_{a \in \mathcal{A}} \lambda_X(a) = 1$. La complexité de cette opération est exactement de q multiplications.

La complexité totale des opérations à l'intérieur d'un noyau est donc de $q(1+\log(q))$ contre seulement 2 opérations pour un décodeur binaire. Notons toutefois que ce facteur $q(1+\log(q))$ ne se retrouve pas nécessairement au niveau de la latence, grâce notamment aux efforts réalisés dans le contexte des codes LDPC, où une réduction significative de celle-ci a pu être obtenu pour les nœuds xor [86].

Sous réserve d'utiliser les fonctions de mises à jour ci-dessus, le processus de décodage est tout à fait similaire à celui d'un code binaire, et les principaux algorithmes de décodages se généralisent naturellement. En dehors du décodage SC, le décodeur par liste (SCL et CA-SCL) est considéré dans [39], tandis que [108] explore à la fois le décodage par propagation de croyance, le décodage à inversion et une technique de réduction de complexité du décodeur par liste.

5.2.3 CONSTRUCTION DU CODE NON-BINAIRE

Pour résoudre le problème de la construction du code polaire, *i.e.* le choix des positions des symboles figés et d'information, les techniques d'évolution de densité ont été généralisées au cas non-binaire dans [39]. Cependant, l'approche la plus simple consiste à utiliser la méthode heuristique, exploitant un décodeur genie-aided, afin de calculer la probabilité d'erreur $p_i^{(ga)} = \Pr(\hat{u}_i \neq u_i \mid u_0^{i-1})$ pour chaque canal virtuel. Les symboles d'information sont alors transmis sur les canaux virtuels ayant la plus faible probabilité d'erreur [24].

Il existe deux possibilités pour positionner les bits d'information:

- Les symboles sont soit figés, soit constitués uniquement de bits d'information. Dans ce cas, un symbole d'information (*i.e.*, constitué uniquement de bits d'information) peut prendre n'importe quelle valeur dans l'alphabet du code $\mathcal{A}$. On note que dans ce cas, le nombre de bits figés (ou d'information) est nécessairement un multiple de p.
- Les symboles peuvent être *partiellement figés*, c'est-à-dire que chaque symbole u_i prend des valeurs dans un sous-ensemble $\mathcal{A}_i \subseteq \mathcal{A}$, avec notamment $|\mathcal{A}_i| = 2^{k_i}$, où $k_i \leq n$ correspond au nombre de bits d'information dans le symbole u_i [24].

Dans la suite de ce chapitre, seule la construction au niveau symbole sera considérée, et l'on supposera que les symboles u_i sont soit figés, soit uniformément distribués sur $\mathcal{A}$.

5.3 Optimisation des permutations du graphe

Pour la suite de cette section, nous introduisons la notion d'*étage* d'un noyau comme le nombre de noyaux entre celui-ci (inclus) et le canal, noté par la lettre $\ell \in \{1, \ldots, n\}$. Ainsi, au niveau du décodeur, les entrées des noyaux du premier étage $(\ell = 1)$ sont obtenues directement à partir des symboles reçus du canal. A l'issue du décodage, les décisions pour les symboles d'information sont notées $\hat{u}_i$.

Nous proposons d'optimiser les permutations du graphe graduellement, en commençant par le premier étage (dans le sens du décodage), puis l'étage suivant et ainsi de suite. Dans un premier temps on présentera le critère d'optimisation. La méthode d'optimisation sera ensuite présentée, d'abord pour les noyaux du premier étage , puis généralisée pour tout noyau du graphe polaire.

5.3.1 Critère d'optimisation

On considère dans cette section un unique noyau tel que présenté dans la figure 5.3, où les symboles d'entrée, supposés uniformément distribués sur $\mathcal{A}$, sont dénommés par u_0 et u_1 et les symboles codés par x_0 et x_1. Pour le décodage, on appelle λ_0, λ_1 les messages d'entrée, λ_X le message en sortie du nœud xor obtenu par l'équation (5.4), et λ_R le message en sortie du nœud répétition, obtenu par l'équation (5.5), en supposant que u_0 est connu (on considère ici le décodeur genie-aided). L'objectif est d'optimiser la permutation π, de manière à minimiser les probabilités d'erreurs en sortie du noyau, lorsqu'un décodeur genie-aided est utilisé. Les estimations des symboles u_0 et u_1 sont notées $\hat{u}_0$ et $\hat{u}_1$ respectivement et sont obtenues d'après:

$$\hat{u}_0 = \arg\max_{a \in \mathcal{A}} \lambda_X(a)$$
$$\hat{u}_1 = \arg\max_{a \in \mathcal{A}} \lambda_R(a)$$

Afin de distinguer entre une variable aléatoire et ses réalisations, il sera utile par la suite de considérer les symboles u_0 et u_1 comme des réalisations de deux variables aléatoires uniformes, notées U_0 et U_1. On notera également $X_0 = U_0 + U_1$ et $X_1 = \pi(U_1)$ et par $\hat{U}_0$ et $\hat{U}_1$ les variables aléatoires associées aux symboles $\hat{u}_0$ et $\hat{u}_1$ respectivement. La probabilité d'erreur sur chaque symbole est définie par:

$$\mathrm{SEP}_X = \Pr(\hat{U}_0 \neq U_0) \tag{5.6}$$
$$\mathrm{SEP}_R = \Pr(\hat{U}_1 \neq U_1 \mid U_0) \tag{5.7}$$

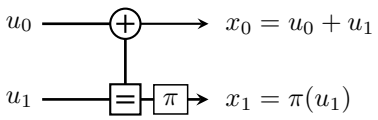


FIGURE 5.3: *Graphe du noyau non-binaire*

Il s'avère qu'en pratique l'impact de la permutation π sur SEP_X est négligeable, en particulier comparé à l'impact sur SEP_R. Notons que:

$$\begin{aligned}\hat{u}_0 &= \underset{a\in\mathcal{A}}{\arg\max}\,\lambda_X(a) \\ &= \underset{a\in\mathcal{A}}{\arg\max} \sum_{\substack{(a_0,a_1)\in\mathcal{A}^2 \\ a_0+\pi^{-1}(a_1)=a}} \lambda_0(a_0)\cdot\lambda_1(a_1)\end{aligned}$$

Cette dernière expression ne peut se simplifier, mais peut être approximée, tout particulièrement pour des valeurs de SNR élevées, par:

$$\sum_{\substack{(a_0,a_1)\in\mathcal{A}^2 \\ a_0+\pi^{-1}(a_1)=a}} \lambda_0(a_0)\cdot\lambda_1(a_1) \approx \max_{\substack{(a_0,a_1)\in\mathcal{A}^2 \\ a_0+\pi^{-1}(a_1)=a}} \lambda_0(a_0)\cdot\lambda_1(a_1)$$

Finalement, on vérifie que:

$$\underset{a\in\mathcal{A}}{\arg\max} \max_{\substack{(a_0,a_1)\in\mathcal{A}^2 \\ a_0+\pi^{-1}(a_1)=a}} \lambda_0(a_0)\cdot\lambda_1(a_1) = \underset{a_0\in\mathcal{A}}{\arg\max}\,\lambda_0(a_0) + \pi^{-1}\left(\underset{a_1\in\mathcal{A}}{\arg\max}\,\lambda_1(a_1)\right) \tag{5.8}$$

Cette dernière expression revient à considérer que le décodeur prend la décision $\hat{u}_0$ à partir des décisions dures sur les entrées du noyaux, notés $\hat{x}_0$ et $\hat{x}_1$ respectivement, et propage alors ces décisions dans le graphe. Les simulations confirment d'ailleurs l'équivalence des performances à partir de cette dernière approximation par rapport au cas d'un décodage souple. On note par $\hat{X}_0$ et $\hat{X}_1$ les variables aléatoires associées à $\hat{x}_0$ et $\hat{x}_1$, et $\mathrm{SEP}_0 = \Pr(\hat{X}_0 \neq X_0)$ et $\mathrm{SEP}_1 = \Pr(\hat{X}_1 \neq X_1)$. Si l'on suppose X_0 et X_1 indépendantes (ce qui toujours le cas pour un décodeur genie-aided, pour autant que cette condition d'indépendance soit vérifiée en entrée du décodeur), le taux d'erreur dans le cas d'un décodage en décision dure, noté $\overline{\mathrm{SEP}_X}$ est donné par:

$$\begin{aligned}\overline{\mathrm{SEP}_X} &= \Pr(\hat{X}_0 \neq X_0, \hat{X}_1 = X_1) + \Pr(\hat{X}_0 = X_0, \hat{X}_1 \neq X_1) \\ &\quad + \Pr(\hat{X}_0 \neq X_0, \hat{X}_1 \neq X_1, \hat{X}_0 + \pi^{-1}(\hat{X}_1) \neq X_0 + \pi^{-1}(X_1)) \\ &= \mathrm{SEP}_0\cdot(1-\mathrm{SEP}_1) + (1-\mathrm{SEP}_0)\cdot\mathrm{SEP}_1 \\ &\quad + \Pr(\hat{X}_0 + \pi^{-1}(\hat{X}_1) \neq X_0 + \pi^{-1}(X_1) \,|\, \hat{X}_0 \neq X_0, \hat{X}_1 \neq X_1)\cdot\mathrm{SEP}_0\cdot\mathrm{SEP}_1 \\ &= \mathrm{SEP}_0 + \mathrm{SEP}_1 - \mathrm{SEP}_0\cdot\mathrm{SEP}_1\cdot[2 - \Pr(\hat{X}_0 + \pi^{-1}(\hat{X}_1) \neq X_0 + \pi^{-1}(X_1) \,|\, \hat{X}_0 \neq X_0, \hat{X}_1 \neq X_1)]\end{aligned}$$

Il apparaît que l'impact de π sur $\overline{\mathrm{SEP}_X}$ est d'autant plus négligeable que les probabilités d'erreurs SEP_0 et SEP_1 sont faibles, du fait que le seul terme dépendant de π est multiplié par un facteur $\mathrm{SEP}_0\cdot\mathrm{SEP}_1$. Ainsi, l'optimisation de la permutation π se résume à trouver celle minimisant la valeur de SEP_R.

La probabilité d'erreur SEP_R peut être efficacement approximée à partir de la borne de l'union (UB):

$$\begin{aligned}
\mathrm{SEP}_R &= \Pr\left(\hat{U}_1 \neq U_1 \,|\, U_0\right) \\
&= \frac{1}{|\mathcal{A}|^2} \sum_{(u_0,u_1)\in\mathcal{A}^2} \Pr\left(\hat{U}_1 \neq u_1 \mid U_0 = u_0, U_1 = u_1\right) \\
&= \frac{1}{|\mathcal{A}|^2} \sum_{(u_0,u_1)\in\mathcal{A}^2} \Pr\left(\underset{u_1'\in\mathcal{A}}{\arg\max}\, \lambda_R(u_1') \neq u_1 \mid U_0 = u_0, U_1 = u_1\right) \\
&\overset{\text{UB}}{\leq} \frac{1}{|\mathcal{A}|^2} \sum_{(u_0,u_1)\in\mathcal{A}^2} \sum_{u_1'\in\mathcal{A}\setminus u_1} \Pr\left(\lambda_R(u_1') \geq \lambda_R(u_1) \mid U_0 = u_0, U_1 = u_1\right)
\end{aligned}$$

On a de plus, en notant $x_0 = u_0 + u_1$, $x_1 = \pi(u_1)$, $x_0' = u_0 + u_1'$, $x_1' = \pi(u_1')$:

$$\begin{aligned}
\Pr\left(\lambda_R(u_1') \geq \lambda_R(u_1) \mid U_0 = u_0, U_1 = u_1\right) &= \Pr(\lambda_0(u_1' + u_0')\lambda_1(\pi(u_1')) \geq \lambda_0(u_1 + u_0)\lambda_1(\pi(u_1)) \\
&\qquad \mid U_0 = u_0, U_1 = u_1) \\
&= \Pr\left(\lambda_0(x_0')\lambda_1(x_1') \geq \lambda_0(x_0)\lambda_1(x_1) \mid X_0 = x_0, X_1 = x_1\right) \\
&\triangleq \Pr\left(x_0', x_1' \,;\, x_0, x_1\right)
\end{aligned}$$

avec $\Pr\left(x_0', x_1' \,;\, x_0, x_1\right)$ correspondant à la probabilité que (x_0', x_1') soit plus probable que (x_0, x_1) alors que $(X_0, X_1) = (x_0, x_1)$. Ainsi, la borne de l'union de SEP_R, notée dans la suite $\overline{\mathrm{SEP}_R^{\pi}}$, s'écrit:

$$\overline{\mathrm{SEP}_R^{\pi}} = \frac{1}{|\mathcal{A}|^2} \sum_{u_0\in\mathcal{A}} \sum_{(x_1,x_1')\in\mathcal{A}_*^2} \Pr\left(u_0 + \pi^{-1}(x_1'), x_1' \,;\, u_0 + \pi^{-1}(x_1), x_1\right) \tag{5.9}$$

où $\mathcal{A}_*^2 = \left\{(x_1, x_1') \in \mathcal{A}^2 \mid x_1 \neq x_1'\right\}$.

Nous proposons d'utiliser le critère de la borne de l'union pour optimiser la permutation π, *i.e.* trouver la permutation Π telle que:

$$\Pi = \underset{\pi\in\mathrm{Sym}(\mathcal{A})}{\arg\min}\, \overline{\mathrm{SEP}_R^{\pi}} \tag{5.10}$$

Pour cela, il est nécessaire d'évaluer au préalable les probabilités $\Pr\left(x_0', x_1'; x_0, x_1\right)$, relatives à l'entrée d'un noyau. Dans un premier temps, on propose de considérer le cas où d'un noyau du premier étage, pour lequel les messages d'entrée sont calculés directement à partir des symboles observés en sortie du canal. Dans ce cas, en supposant un canal AWGN, les valeurs $\Pr\left(x_0', x_1'; x_0, x_1\right)$ se calculent analytiquement à partir de la distance euclidienne sur la constellation utilisée. Dans un deuxième temps, on considérera un noyau d'un étage quelconque du graphe polaire. Les probabilités seront alors évaluées par simulations, et la permutation Π sera obtenue d'après ces dernières.

5.3.2 Optimisation des noyaux sur le premier étage

On considère le cas d'un canal AWGN dont l'alphabet d'entrée est donnée par une constellation $\mathcal{X}$, de dimension $|\mathcal{X}| = 2^m$, telle que $m|p$[1]. On considérera que $\mathcal{X}$ peut être aussi bien réelle que complexe, et l'on écrira $\mathcal{X} \subset \mathbb{K}$, avec soit $\mathbb{K} = \mathbb{R}$ ou $\mathbb{K} = \mathbb{C}$. On considère de plus une fonction d'attribution:

$$\varphi : \mathbb{F}_2^m \to \mathcal{X},$$

[1]En pratique, les cas auxquels on intéressera sont celui d'un canal à entrée binaire, donc $m = 1$, ou celui d'un canal non-binaire avec $m = p$

associant un symbole de la constellation à chaque séquence de m bits. De plus, pour toute séquence $(c_1, c_2, \ldots, c_t) \in \mathbb{F}_2^{mt}$, où $c_i \in \mathbb{F}_2^m$, $i = 1, \ldots, t$, on définit:

$$\varphi(c_1, c_2, \ldots, c_t) = (\varphi(c_1), \varphi(c_2), \ldots, \varphi(c_t)) \in \mathcal{X}^t$$

En particulier, puisque $\mathcal{A} = \mathbb{F}_2^p = (\mathbb{F}_2^m)^{p/m}$, pour tout $x_1, x_2 \in \mathcal{A}$, on écrira $\varphi(x_1), \varphi(x_2) \in \mathcal{X}^{p/m}$ et $\varphi(x_1, x_2) = (\varphi(x_1), \varphi(x_2)) \in \mathcal{X}^{2p/m}$. Pour $(x_1', x_2') \in \mathcal{A}^2$ et $(x_1, x_2) \in \mathcal{A}^2$, on définit également:

$$D_{\mathcal{E}}(x_1', x_2'; x_1, x_2) = \|\varphi(x_1', x_2') - \varphi(x_1, x_2)\|,$$

c'est-à-dire la distance euclidienne (dans $\mathbb{K}^{2p/m}$) entre $\varphi(x_1', x_2')$ et $\varphi(x_1, x_2)$.

Proposition 22 *Pour tout noyau du premier étage, et en supposant un canal AWGN, la probabilité* $\overline{SEP_R^{\pi}}$ *s'écrit:*

$$\overline{SEP_R^{\pi}} = \frac{1}{|\mathcal{A}|^2} \sum_{u_0 \in \mathcal{A}} \sum_{(u_1, u_1') \in \mathcal{A}_*^2} Q\left(\frac{\sqrt{\kappa}}{2\sqrt{2}\sigma} D_{\mathcal{E}}(u_0 + u_1', \pi(u_1'); u_0 + u_1, \pi(u_1)) \right) \tag{5.11}$$

où Q est reliée à la fonction d'erreur gaussienne complémentaire par $Q(x) = \frac{1}{2} erfc(\frac{x}{\sqrt{2}})$.

Cette proposition est démontrée en Annexe D. Notons que la distance euclidienne correspond à:

$$D_{\mathcal{E}}(u_0 + a, \pi(a); u_0 + b, \pi(b)) = \sqrt{\|\varphi(a + u_0) - \varphi(b + u_0)\|^2 + \|\varphi(\pi(a)) - \varphi(\pi(b))\|^2}$$

En définissant l'ensemble des valeurs des distances euclidiennes par:

$$\mathcal{D} = \{ \sqrt{\|\varphi(a + u_0) - \varphi(b + u_0)\|^2 + \|\varphi(\pi(a)) - \varphi(\pi(b))\|^2} \mid (a, b, u_0) \in \mathcal{A}_*^2 \times \mathcal{A} \}, \tag{5.12}$$

et les multiplicités associées :

$$S^{(\pi)}(d) = |\{\|\varphi(a + u_0) - \varphi(b + u_0)\|^2 + \|\varphi(\pi(a)) - \varphi(\pi(b))\|^2 = d^2 \mid (a, b, u_0) \in \mathcal{A}_*^2 \times \mathcal{A}\}|, \tag{5.13}$$

la borne de l'union peut se réécrire:

$$\overline{\mathrm{SEP}_R^{\pi}} = \frac{1}{|\mathcal{A}|^2} \sum_{d \in \mathcal{D}} S^{(\pi)}(d) \cdot Q\left(\sqrt{\kappa} \frac{d}{2\sqrt{2}\sigma} \right) \tag{5.14}$$

Il apparaît qu'un critère possible pour optimiser la permutation π est de maximiser la distance euclidienne minimale définie par:

$$D_{min}^{(\pi)} = \min_{(a, b, u_0) \in \mathcal{A}_*^2 \times \mathcal{A}} \sqrt{\|\varphi(a + u_0) - \varphi(b + u_0)\|^2 + \|\varphi(\pi(a)) - \varphi(\pi(b))\|^2} \tag{5.15}$$

et de minimiser la multiplicité $S^{(\pi)}(D_{min}^{(\pi)})$.

Notons que dans le cas d'un canal à entrée binaire ($\mathcal{X} = \{-1, +1\}$), la distance minimale s'écrit en fonction de la distance de Hamming:

$$D_{min}^{(\pi)} = 2 \min_{(a, b) \in \mathcal{A}_*^2} \sqrt{|a + b|^2 + |\pi(a) + \pi(b)|^2}, \tag{5.16}$$

où $|a|$ correspond au poids de Hamming du vecteur a. Il s'avère finalement qu'optimiser la permutation π revient, pour un canal à entrée binaire, à optimiser l'image binaire du code, tandis que pour un canal non-binaire, cela revient à optimiser la distance euclidienne des mots de code sur la constellation. Ces

deux approches ont été toute deux considérées pour le cas des codes LDPC dans respectivement [75] et [2].

Notons que, pour un canal non-binaire, la distance minimale dépend à la fois de π et de la fonction d'attribution ϕ. Dans la suite, on considérera que la fonction d'attribution ϕ est fixée et correspond à la fonction d'attribution de Gray.

Pour trouver la permutation Π optimale du point de vue de la distance minimale et de la multiplicité associée, la méthode naïve consisterait à passer en revue l'ensemble des $2^p!$ permutations, ce qui devient très rapidement impraticable pour des valeurs de $p \geq 4$. Nous proposons plutôt un algorithme pour construire graduellement une permutation dont la distance minimale est supérieure à un seuil donné, noté D_{th}. La recherche étant d'autant plus rapide que le seuil est élevé, la solution optimale peut être efficacement obtenue en initialisant le seuil à une valeur élevée, puis à le réduire tant qu'aucune solution n'est trouvée. Cet algorithme est applicable aussi bien pour les canaux à entrée binaire que non-binaire.

L'algorithme proposé s'inspire de la méthode par *séparation et évaluation* (ou *branch and bound* en anglais). La première étape, appelée *séparation*, consiste à diviser le problème d'optimisation en un certains nombre de sous-problèmes dont la solution peut être obtenue efficacement. Pour le cas qui nous occupe, il s'agit de concevoir une permutation π comme un tableau indiquant, pour chaque symbole $a \in \mathcal{A}$, l'image de a par π. Dès lors, le problème se décompose à partir de tableaux $\tilde{\pi}$, dont seulement une partie des valeurs sont définies (on notera $\tilde{\pi}(a) = \emptyset$ pour indiquer que l'image du symbole a n'est pas définie), et pour simplifier, on considérera que le tableau est rempli dans l'ordre (en fixant une fois pour toute un ordre quelconque sur $\mathcal{A}$). Dès lors, il est possible de construire un arbre complet des tableaux possibles pour $\tilde{\pi}$, dont la racine est le tableau $\pi(a) = \emptyset, \forall a \in \mathcal{A}$, la profondeur est q, et les *feuilles* correspondent à l'ensemble des permutations π possibles. La deuxième étape, appelée l'*évaluation*, consiste à associer à chaque nœud de l'arbre un *coût*. La fonction de coût pour le problème considéré consiste à évaluer:

$$D_{min}^{(\tilde{\pi})} = \min_{\substack{(a,b)\in\mathcal{A}_*^2 \\ \tilde{\pi}(a)\neq\emptyset, \tilde{\pi}(b)\neq\emptyset}} \left(\min_{u_0\in\mathcal{A}} \|\varphi(a+u_0) - \varphi(b+u_0)\|^2 \right) + \|\varphi(\pi(a)) - \varphi(\pi(b))\|^2 \tag{5.17}$$

Nécessairement, $D_{min}^{(\tilde{\pi})} \geq D_{min}^{(\pi)}$, où π est une feuille descendante du nœud $\tilde{\pi}$, de sorte que rechercher les solutions telles que $D_{min}^{(\pi)} \geq D_{th}$ impose d'avoir également $D_{min}^{(\tilde{\pi})} \geq D_{th}$. Ainsi, dès lors qu'un nœud de l'arbre est tel que $D_{min}^{(\tilde{\pi})} < D_{th}$, il est inutile d'explorer cette branche plus en profondeur. Finalement, l'algorithme proposé consiste en une exploration exhaustive des branches de l'arbre tant que la valeur de la fonction de coût reste supérieure à D_{th}. Cet algorithme permet de garantir l'optimalité du résultat tout en offrant une complexité radicalement inférieure à celle d'un exploration exhaustive.

5.3.2.1 Cas d'un canal à entrée binaire

Proposition 23 *Soit $\pi \in Sym(\mathcal{A})$ et $D_{min}^{(\pi)}$ tel que défini à l'équation (5.16). Alors*

$$\begin{aligned} D_{min}^{(\pi)} &\leq \sqrt{2}p, && \text{si } p \text{ est pair} \\ D_{min}^{(\pi)} &\leq \sqrt{2}\sqrt{p^2+1}, && \text{si } p \text{ est impair} \end{aligned}$$

La démonstration est donnée en Annexe D.

Nous avons déterminé les permutations optimales (*i.e.*, maximisant la distance minimale et minimisant la multiplicité de celle-ci) pour $p = 2,3,4$, et pour chacun des ensembles $\mathrm{GF}(\mathcal{A}), \mathrm{GL}(\mathcal{A}), \mathrm{Sym}(\mathcal{A})$. Pour

$\mathrm{GF}(\mathcal{A})$ et $\mathrm{GL}(\mathcal{A})$ nous avons procédé par recherche exhaustive, alors que pour $\mathrm{Sym}(\mathcal{A})$ nous avons utilisé la méthode par séparation et évaluation décrite dans la section précédente. Notons également que $\mathrm{GF}(\mathcal{A}) \subset \mathrm{GL}(\mathcal{A}) \subset \mathrm{Sym}(\mathcal{A})$, et que pour chacun de ces ensembles plusieurs permutations peuvent correspondre à la solution optimale en termes de distance minimale et multiplicité. Nous nous contenterons néanmoins d'indiquer une seule permutation optimale par ensemble, voire une permutation optimale pour plusieurs ensembles, lorsque ces ensembles ont les mêmes valeurs optimales ($D_{min}^{(\pi)}$, $S^{(\pi)}(D_{min}^{(\pi)})$), auquel cas la permutation indiquée appartiendra au plus petit des ensembles. Les résultats sont présenté dans le tableau 5.1. Les permutations optimales y sont indiquées en identifiant un symbole $a = (a(0),...,a(p-1)) \in \mathcal{A} = \mathbb{F}_2^p$, à l'entier $\sum_{i=0}^{p-1} a(i) \cdot 2^i$.

- Pour $p = 2$ et $p = 3$, les trois ensembles ont la même solution optimale ($D_{min}^{(\pi)}$, $S^{(\pi)}(D_{min}^{(\pi)})$). Les permutations indiquées appartiennent à $\mathrm{GF}(\mathcal{A})$.
- Pour $p = 4$, la borne supérieure de la Proposition 24 ne peut pas être obtenue dans $\mathrm{GF}(\mathcal{A})$, mais il existe toujours une solution linéaire atteignant cette borne et ayant d'ailleurs la même multiplicité que la meilleure solution non-linéaire.

Notons que pour les valeurs de p considérées dans ce tableau, la borne sur la distance minimale définie dans la Proposition 24 est effectivement atteinte dans $\mathrm{GF}(\mathcal{A})$ ou $\mathrm{GL}(\mathcal{A})$ (et a fortiori dans $\mathrm{Sym}(\mathcal{A})$), mais ce résultat n'est pas garanti pour toute valeur de p.

Les trois permutations linéaires, pour $p = 2$, $p = 3$ et permutation dans $\mathrm{GL}(\mathcal{A})$ pour $p = 4$, correspondent aux trois matrices ci-dessous:

$$H_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, H_3 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ et } H_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

La figure 5.4 présente, pour $p = 4$, les résultats de simulation pour la valeur de SEP_R avec les permutations optimales dans $\mathrm{GF}(\mathcal{A})$ et $\mathrm{GL}(\mathcal{A})$. Cela permet de confirmer que les performances obtenues sont fortement dépendantes de la distance minimale et sa multiplicité. La perte en termes de distance minimale lorsque l'on se restreint à $\mathrm{GF}(\mathcal{A})$ se traduit par une détérioration des performances, s'élevant à près de 0.5dB pour un taux d'erreur de 10^{-3}.

TABLE 5.1: *Distances minimales et multiplicités des permutations pour un canal à entrée binaire*

p	Ensemble de définition de π	Permutation optimale	$D_{min}^{(\pi)}$	$S^{(\pi)}(D_{min}^{(\pi)})$
2	$\mathrm{GF}(\mathcal{A})$,$\mathrm{GL}(\mathcal{A})$,$\mathrm{Sym}(\mathcal{A})$	[0,3,2,1]	$2\sqrt{2}$	4
3	$\mathrm{GF}(\mathcal{A})$,$\mathrm{GL}(\mathcal{A})$,$\mathrm{Sym}(\mathcal{A})$	[0,6,7,1,5,3,2,4]	$2\sqrt{5}$	32
4	$\mathrm{GF}(\mathcal{A})$	[0,10,7,13,14,4,9,3,15,5,8,2,1,11,6,12]	$2\sqrt{5}$	64
	$\mathrm{GL}(\mathcal{A})$, $\mathrm{Sym}(\mathcal{A})$	[0,14,13,3,11,5,6,8,7,9,10,4,12,2,1,15]	$4\sqrt{2}$	96

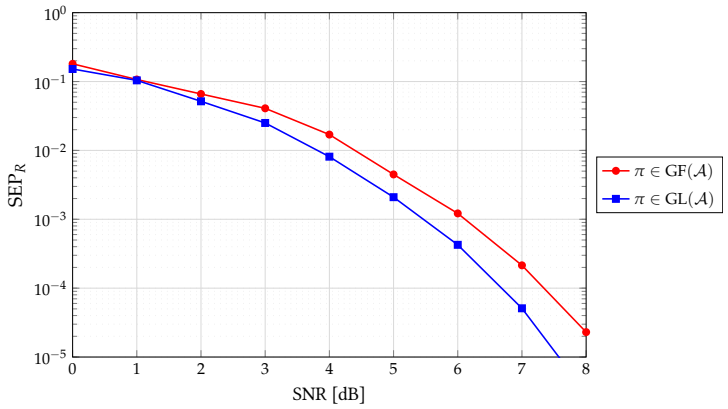


FIGURE 5.4: *Impact de la permutation π sur la valeur de SEP_R pour $p = 4$ (BI-AWGN)*

5.3.2.2 CAS D'UN CANAL À ENTRÉE NON-BINAIRE

On s'intéresse maintenant à un canal non-binaire, et l'on compare la meilleure solution en terme de distance minimale obtenue par l'algorithme proposé (donc dans $\mathrm{Sym}(\mathcal{A})$) aux meilleures solutions dans $\mathrm{GF}(\mathcal{A})$ et $\mathrm{GL}(\mathcal{A})$ respectivement, obtenues par recherche exhaustive. Le tableau 5.2 résume les valeurs de distances minimales et multiplicités obtenues pour $p = 4$ avec une constellation 16-QAM et le codage de Gray comme fonction d'attribution (Chapitre 4, Figure 4.3). Il apparaît une nouvelle fois que la restriction à $\mathrm{GF}(\mathcal{A})$ induit une perte de distance minimale, tandis que $\mathrm{GL}(\mathcal{A})$ permet d'obtenir une solution équivalente à la meilleure permutation dans $\mathrm{Sym}(\mathcal{A})$ du point de vue de la distance minimale comme de la multiplicité. Il a été observé qu'il n'existe que deux matrices permettant d'obtenir cette distance minimale et cette multiplicité, données par:

$$H_4' = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \text{ et } H_4'' = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Nous avons également envisagé le cas où la valeur de u_0 n'est pas distribuée uniformément sur $\mathcal{A}$ mais figée à la valeur 0 [2]. Dans ce cas, par contre, il s'avère que la meilleure solution dans $\mathrm{GL}(\mathcal{A})$ n'atteint pas la valeur de distance minimale de la meilleure solution dans $\mathrm{Sym}(\mathcal{A})$. La figure 5.5 présente la valeur du SEP_R obtenue par simulation. Tout comme pour un canal binaire, une perte significative est induite par la restriction à l'ensemble $\mathrm{GF}(\mathcal{A})$, s'élevant à près de 1dB pour un taux d'erreur de 10^{-4}. La performance de la meilleure solution dans $\mathrm{Sym}(\mathcal{A})$ lorsque $u_0 = 0$ est également montrée. Afin de ne pas surcharger la figure, les performances lorsque $u_0 = 0$ dans $\mathrm{GL}(\mathcal{A})$ et dans $\mathrm{GF}(\mathcal{A})$ n'ont pas été tracées dans la mesure où elles se confondent avec le cas où $u_0 \in \mathcal{A}$. Dès lors, il apparaît qu'un gain de 0.2dB peut être obtenu lorsque $u_0 = 0$ dans $\mathrm{Sym}(\mathcal{A})$ par rapport à $\mathrm{GL}(\mathcal{A})$.

[2] Un tel cas de figure pourrait se présenter lorsque l'on sait que le symbole u_0 correspond à un symbole figé, ou dépend uniquement de symboles figés.

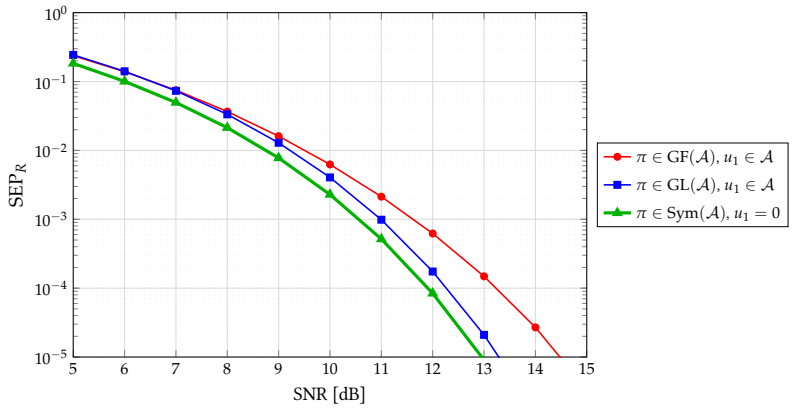


FIGURE 5.5: *Impact de la permutation π sur la valeur de SEP_R pour une constellation 16-QAM (BI-AWGN)*

TABLE 5.2: *Distances minimales et multiplicités des permutations pour la modulation 16-QAM*

u_0	Ensemble de définition de π	Permutation optimale	$(D^{\pi}_{\min})^2$	$S^{(\pi)}(D^{(\pi)}_{min})$
$u_0 \in \mathcal{A}$	$\mathrm{GF}(\mathcal{A})$	$[0,3,6,5,12,15,10,9,11,8,13,14,7,4,1,2]$	1.2	128
$u_0 \in \mathcal{A}$	$\mathrm{GL}(\mathcal{A}), \mathrm{Sym}(\mathcal{A})$	$[0,12,13,1,3,15,14,2,7,11,10,6,4,8,9,5]$	1.6	16
$u_0 = 0$	$\mathrm{GF}(\mathcal{A})$	$[0,3,6,5,12,15,10,9,11,8,13,14,7,4,1,2]$	1.2	8
$u_0 = 0$	$\mathrm{GL}(\mathcal{A})$	$[0,12,13,1,3,15,14,2,7,11,10,6,4,8,9,5]$	1.6	2
$u_0 = 0$	$\mathrm{Sym}(\mathcal{A})$	$[15,12,13,14,1,2,3,0,7,4,5,6,9,10,11,8]$	2.0	32

5.3.3 CAS GÉNÉRAL

Si un gain notable peut être obtenu pour les noyaux du premier étage du graphe polaire, il s'avère cependant que celui-ci ne se retrouve pas au niveau du taux d'erreur du code non-binaire si les permutations des étages suivants ne sont pas également optimisées. Cela constitue l'objet de cette section, à savoir, généraliser la méthode proposée pour le premier étage à tout étage du graphe. Notons que si l'on suppose que les symboles transmis sont uniformément distribués sur $\mathcal{A}$, alors le nombre de permutations à optimiser est réduit significativement. En effet, considérons le graphe polaire présenté dans la figure 5.6, et intéressons-nous tout d'abord aux $N/2$ noyaux du premier étage. Tous ces noyaux ont les mêmes distributions en entrée. Dès lors, la même permutation $\pi_1^{(1)}$ est appliquée à chaque fois. De manière générale, le graphe d'un code polaire de longueur N s'obtient à partir de deux graphes de codes non-binaires de longueurs $N/2$ (en rouge sur la figure), associés par un ultime étage de noyaux. Ainsi, les deux sous-blocs de longueur $N/2$ jouent des rôles symétriques et "voient" le même canal, de sorte que les permutations optimales obtenues pour l'un s'appliquent aussi bien à l'autre. On obtient finalement qu'il suffit d'optimiser exactement $\sum_{i=0}^{n-1} 2^i = N-1$ permutations du graphe.

Cette remarque sur la réduction de nombre de permutations à optimiser a également son importance pour le processus d'optimisation des permutations. Les permutations vont en effet être optimisées étage par étage, en commençant par le tout premier pour finir avec l'étage $\ell = n$. Pour chaque permutation à optimiser, on verra qu'il est nécessaire d'évaluer de manière heuristique certaines probabilités relatives aux entrées (dans le sens du décodage) du noyau concerné. Pour cela, il n'est cependant pas nécessaire de

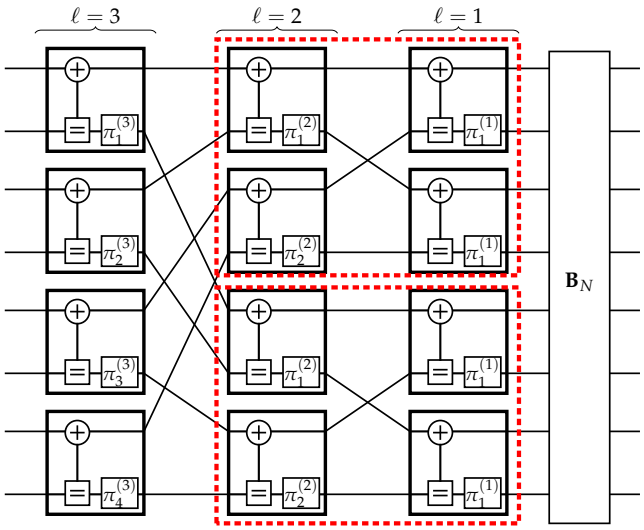


FIGURE 5.6: *Optimisation des permutations dans le graphe* $N = 8$

procéder à un encodage-décodage sur tout le graphe du code non-binaire. Considérons, par exemple, que l'on souhaite optimiser les permutations de l'étage $\ell = 3$ du graphe représenté sur la 5.6. On remarque d'abord que les deux entrées de chaque noyau correspondent à deux sorties appariées de deux codes non-binaires identiques, de longueur $N' = 4$ (encadrés en rouge sur la figure). Pour obtenir les propriétés statistiques de celles-ci, il est alors suffisant de réaliser des simulations Monte-Carlo uniquement sur le code non-binaire de longueur $N' = 4$. Plus généralement, pour obtenir les entrées d'un noyau de l'étage ℓ, il est suffisant d'utiliser le code non-binaire de longueur $N' = 2^{\ell-1}$.

En général il n'est pas possible d'obtenir simplement les distributions en entrée d'un noyau autre que sur le premier étage. L'exception à cette règle concerne les permutations apparaissant sur le dernière ligne du graphe ($\pi_4^{(3)}, \pi_2^{(2)}, \pi_1^{(1)}$ sur la figure 5.6). Dans ce cas particulier, il peut être montré que $\overline{\text{SEP}}_R^{\pi}$ peut également s'exprimer en fonction des distances euclidiennes sur la constellation. Toutefois, il n'est pas praticable d'optimiser toutes ces permutations conjointement. Soulignons de plus qu'une optimisation conjointe améliorerait la probabilité d'erreur sur le tout dernier symbole $\hat{u}_{N-1}$, mais risque de détériorer celles des symboles précédents, et est donc susceptible d'engendrer en réalité une perte sur le taux d'erreur global.

Rappelons l'expression de la borne de l'union du taux d'erreur pour un noyau:

$$\overline{\text{SEP}}_R^{\pi} = \frac{1}{|\mathcal{A}|^2} \sum_{u_0 \in \mathcal{A}} \sum_{(u_1, u_1') \in \mathcal{A}_*^2} \Pr\left(u_0 + u_1', \pi(u_1'); u_0 + u_1, \pi(u_1)\right) \tag{5.18}$$

avec $\Pr(x_0', x_1'; x_0, x_1)$ correspondant à la probabilité que (x_0', x_1') soit plus probable que (x_0, x_1) alors que $(x_0, x_1) = (X_0, X_1)$. A défaut de relier la probabilité $\overline{\text{SEP}}_R^{\pi}$ aux distances euclidiennes, il est suffisant de disposer des probabilités $\Pr(x_0', x_1'; x_0, x_1)$ pour tout quadruplet $(x_0', x_1', x_0, x_1) \in \mathcal{A}^4$ et l'on va montrer que ces probabilités sont suffisantes pour optimiser la permutation π. Tout d'abord, notons qu'il est

Algorithme 8 Calcul des probabilités $P_M(u_1 + u_1', \pi(u_1'), \pi(u_1))$

```
1:  procédure CalculPm
2:      Initialiser P_M(x, x'_1, x_1) = 0;
3:      pour (x_0, x_1) ∈ A^2 faire
4:          pour t = 0 : T − 1 faire
5:              Encoder (v_0, ..., v_{k−1}, x_0, x_1, v_{k+2}, ..., v_{2^{ℓ−1}});
6:              Décoder et obtenir λ_0 et λ_1;
7:              pour (x, x'_1) ∈ A^2 faire
8:                  si λ_0(x + x_0) · λ(x'_1) ≤ λ_0(x_0) · λ_1(x_1) alors
9:                      P_M(x, x'_1, x_1) ← P_M(x, x'_1, x_1) + 1;
10:                 fin si
11:             fin pour
12:         fin pour
13:     fin pour
14:     P_M = P_M / T;
15: fin procédure
```

possible de réduire le nombre de probabilités nécessaires de q^4 à q^3 de la manière suivante:

$$\begin{aligned}
\overline{\mathrm{SEP}_R^{\pi}} &= \frac{1}{|\mathcal{A}|^2} \sum_{u_0 \in \mathcal{A}} \sum_{(u_1, u_1') \in \mathcal{A}_*^2} \Pr\left(u_0 + u_1', \pi(u_1') \,;\, u_0 + u_1, \pi(u_1)\right) \\
&= \frac{1}{|\mathcal{A}|^2} \sum_{(u_1, u_1') \in \mathcal{A}_*^2} \sum_{u_0 \in \mathcal{A}} \Pr\left(u_0 + u_1', \pi(u_1') \,;\, u_0 + u_1, \pi(u_1)\right) \\
&= \frac{1}{|\mathcal{A}|^2} \sum_{(u_1, u_1') \in \mathcal{A}_*^2} P_M(u_1 + u_1', \pi(u_1'), \pi(u_1)) \qquad (5.19)
\end{aligned}$$

avec :

$$P_M(u_1 + u_1', \pi(u_1'), \pi(u_1)) = \sum_{u_0 \in \mathcal{A}} \Pr\left(u_0 + u_1', \pi(u_1') \,;\, u_0 + u_1, \pi(u_1)\right) \qquad (5.20)$$

Les probabilités $P_M(u_1 + u_1', \pi(u_1'), \pi(u_1))$ sont calculées de manière heuristique par l'algorithme 8 pour chaque noyau dont on cherche à optimiser la permutation. Supposons un noyau de l'étage $\ell \in \{0, \ldots, n-1\}$. Il s'agit de générer les symboles (x_0, x_1) uniformément sur $\mathcal{A}^2$ un nombre $q^2 \cdot T$ de fois (T est supposé suffisamment large pour que les probabilités obtenues soient fiables), de les encoder dans un code non-binaire de longueur $2^{\ell-1}$, pour lequel les autres symboles (notés v_i dans l'algorithme 8) sont générés aléatoirement et uniformément sur $\mathcal{A}$. On procède ensuite au décodage genie-aided de manière à obtenir les messages en entrée du noyau considéré, notés λ_0 et λ_1, pour finalement incrémenter la valeur de $P_M(x, x_1', x_1)$ dans les cas où $(x + x_0, x_1')$ est plus probable que (x_0, x_1).

Finalement, à partir de ces probabilités, nous expliquons comment optimiser la permutation π. Le principe est tout à fait similaire à l'algorithme proposé pour le premier étage, à savoir une méthode par *séparation et évaluation*, où la permutation est construite graduellement. La différence vient du fait que la fonction d'évaluation ne peut plus être la distance minimale, mais nous proposons d'utiliser directement la valeur de $\overline{\mathrm{SEP}_R^{\pi}}$, calculée d'après la formule (5.19) et des probabilités P_M obtenues préalablement. Pour toute branch de l'arbre, associé à une permutation $\tilde{\pi}$ (partiellement définie ou non), on définit la

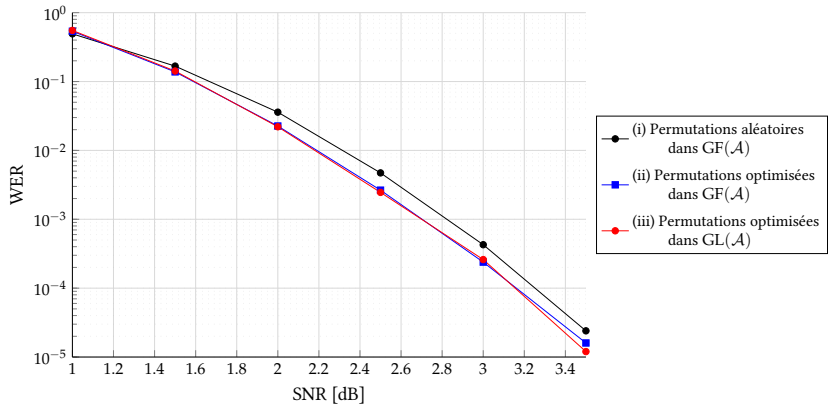


FIGURE 5.7: *Impact du choix des permutations du graphe ($p = 4, N = 256, R = 1/2$, BI-AWGN)*

probabilité $\widetilde{\text{SEP}_R}$ par:

$$\widetilde{\text{SEP}_R} = \frac{1}{|\mathcal{A}|^2} \sum_{\substack{(u_1,u_1')\in\mathcal{A}_*^2 \\ \tilde{\pi}(u_1)\neq\emptyset,\tilde{\pi}(u_1')\neq\emptyset}} P_M(u_1 + u_1', \pi(u_1'), \pi(u_1)) \tag{5.21}$$

Lorsque cette probabilité dépasse un seuil fixé SEP_{th}, l'algorithme n'explorera pas la branche de l'arbre plus en profondeur, du fait que que $\widetilde{\text{SEP}_R} \leq \overline{\text{SEP}_R^{\pi}}$, de sorte que le taux d'erreur $\overline{\text{SEP}_R^{\pi}}$ pour tous les nœuds descendants de la branche considérée auront nécessairement un taux d'erreur supérieur au seuil.

Il s'avère cependant que cet algorithme est excessivement couteux en termes de ressources de calculs. Hormis pour les codes courts, où le nombre de permutations à déterminer est limité, l'optimisation des permutations dans $\text{Sym}(\mathcal{A})$ reste difficilement abordable. Une manière de simplifier la complexité d'un tel algorithme consiste à se restreindre à $\text{GL}(\mathcal{A})$ voire $\text{GF}(\mathcal{A})$, où il est suffisant de calculer la valeur de $\overline{\text{SEP}_R^{\pi}}$ pour chaque $\pi \in \text{GL}(\mathcal{A})$ ou $\pi \in \text{GF}(\mathcal{A})$ respectivement, à partir des probabilités P_M. Pour rester dans $\text{Sym}(\mathcal{A})$ tout en s'efforçant de réduire la complexité, une méthode pourrait être de considérer un algorithme par liste ou un algorithme de Stack.

5.4 RÉSULTATS DE SIMULATIONS

La méthode proposée pour optimiser les permutations dans $\text{Sym}(\mathcal{A})$ nécessite un long processus d'optimisation, c'est pourquoi nous nous contenterons ici de discuter de l'optimisation des permutations dans $\text{GF}(\mathcal{A})$ et $\text{GL}(\mathcal{A})$.

On considère un code non-binaire, avec $p = 4$, de longueur $N = 2^8$ symboles avec $K = 512$ bits d'informations (rendement $512/(pN) = 1/2$). La figure 5.7 montre les performances sur un canal BI-AWGN. Les trois solutions comparées sont les suivantes:

(i) pour chaque noyau, une permutation différente est choisie aléatoirement dans $\text{GF}(\mathcal{A})$.

(ii) pour chaque noyau, la permutation est optimisée dans $\text{GF}(\mathcal{A})$, en utilisant la méthode proposée.

(iii) pour chaque noyau, la permutation est optimisée dans $\text{GL}(\mathcal{A})$, en utilisant la méthode proposée.

Les simulations montrent que le fait d'optimiser les permutations dans $\text{GF}(\mathcal{A})$ ou $\text{GL}(\mathcal{A})$ fournit un avantage léger mais clair par rapport au cas où chaque permutation est choisie aléatoirement dans $\text{GF}(\mathcal{A})$. Par contre, la différence entre l'optimisation dans $\text{GF}(\mathcal{A})$ et dans $\text{GL}(\mathcal{A})$ s'avère très limitée.

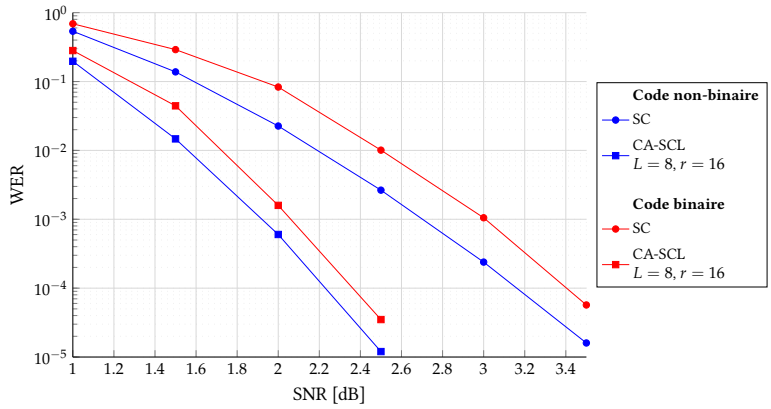


FIGURE 5.8: *Comparaison des décodeurs binaire et non-binaire ($p = 4, N = 256, R = 1/2$, BI-AWGN)*

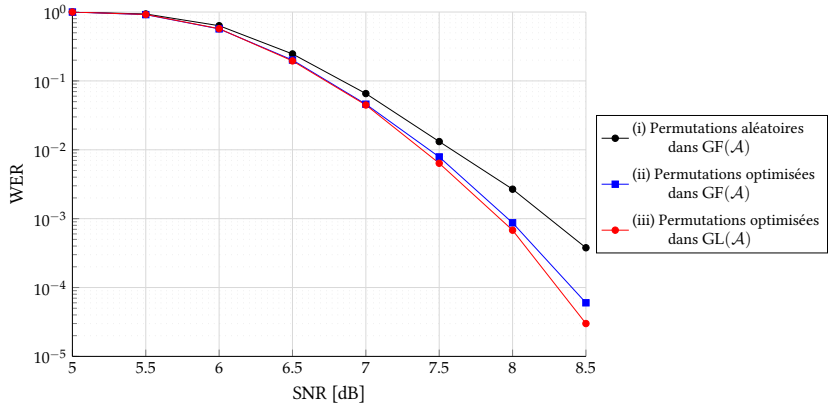


FIGURE 5.9: *Impact du choix des permutations du graphe ($p = 4, N = 256, R = 1/2$, 16-QAM, AWGN)*

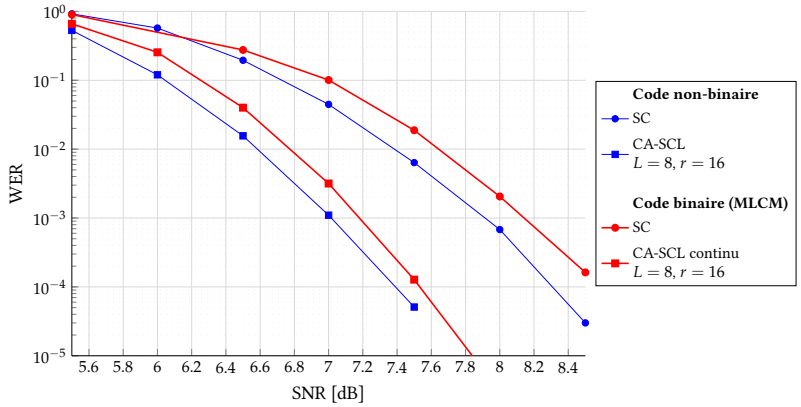


FIGURE 5.10: *Comparaison du décodage non-binaire et du décodage binaire MLCM ($p = 4, N = 256, R = 1/2$, 16-QAM, AWGN)*

La figure 5.8 utilise exactement le même code et les mêmes paramètres que précédemment, mais compare les performances entre un code binaire et le code non-binaire dont les permutations sont optimisées dans GL($\mathcal{A}$). En particulier, un décodeur CA-SCL avec $L = 8$ et 16 bits de CRC est employé. Le code binaire utilisé pour la comparaison est un code de longueur $N = 1024$ et $K = 512$. Il apparait que le décodeur non-binaire supplante assez nettement le décodeur binaire, tant pour le décodeur SC que CA-SCL.

Les figures 5.9 et 5.10 fournissent les mêmes observations avec une modulation 16-QAM. Là encore,il apparait qu'optimiser les permutations dans GF($\mathcal{A}$) ou GL($\mathcal{A}$) apporte un avantage tangible par rapport

au cas où les permutations sont choisies aléatoirement, mais que la différence entre $\mathrm{GF}(\mathcal{A})$ et $\mathrm{GL}(\mathcal{A})$ est très limitée. Qui plus est, la figure 5.10 met en évidence l'avantage des codes non-binaires avec permutations optimisées dans $\mathrm{GL}(\mathcal{A})$ par rapport à la modulation MLCM, aussi bien avec le décodeur SC que CA-SCL. Soulignons que les codes non-binaires considérés maintiennent un gain de 0.2dB par rapport à la meilleure solution du Chapitre 4, à savoir le MLCM avec un décodeur par liste continu.

5.5 Conclusion

Ce chapitre a abordé la question des codes polaires non-binaires, définis récursivement à partir de la transformation élémentaire $(u_0, u_1) \to (u_0 + u_1, \pi(u_1))$. La question du choix des permutations du graphe a été abordée et une méthode a été proposée afin d'optimiser ces permutations pour tout noyau du graphe. Pour les noyaux situés directement après le canal, l'optimisation revient à optimiser l'image binaire du code pour un canal à entrée binaire, ou à optimiser la distance euclidienne sur la constellation pour un canal non-binaire. Il a été mis en évidence qu'une optimisation précise de la permutation pour un noyau du premier étage pouvait apporter un gain significatif. Néanmoins, il apparaît qu'une optimisation tout aussi fine des permutations des étages suivants est également nécessaire afin de capitaliser sur l'avantage obtenu sur ce premier étage. La méthode proposée est étendue à tous les étages du graphe, où la distance euclidienne est remplacée par une matrice de probabilités, calculées de manière heuristique. Si la méthode proposée exige un long processus d'optimisation pour le cas où la permutation est définie dans $\mathrm{Sym}(\mathcal{A})$, et elle reste parfaitement praticable lorsque l'on se restreint à $\mathrm{GF}(\mathcal{A})$ ou $\mathrm{GL}(\mathcal{A})$.

Les résultats dans $\mathrm{GF}(\mathcal{A})$ et $\mathrm{GL}(\mathcal{A})$ montrent que l'optimisation des permutations offre un gain léger mais clair par rapport au choix aléatoire de chaque permutation du graphe, aussi bien pour un canal binaire que non-binaire. De plus, il a été mis en évidence le fait que les codes non-binaires apportent un avantage significatif par rapport à leurs homologues binaires.

Les résultats présentés dans ce chapitre considèrent le cas où $p = 4$, pour lequel l'analyse exhaustive dans $\mathrm{GL}(\mathcal{A})$ est d'une complexité abordable. Pour des valeurs de p supérieure, une méthode sous-optimale permettant de réduire le nombre de permutations considérées est requise. L'utilisation d'un algorithme de Stack apparaît comme une solution possible.

Conclusion

Depuis leur découverte en 2008 par Arikan, les codes polaires ont connu un essor excessivement rapide jusqu'à leur récente adoption dans les standards de la 5G. Si cet engouement initial s'explique par leur aptitude à atteindre asymptotiquement la capacité de Shannon, grâce au phénomène de polarisation, les années qui suivirent ont révélé leur attractivité face à une large variété de problématiques rencontrées. Cette thèse s'est concentrée autour de deux problématiques complémentaires concernant les codes polaires. D'une part, la construction des codes est investiguée du point de vue de l'optimisation du spectre des distances, de l'utilisation du poinçonnage ou du raccourcissement et finalement dans le contexte des modulations d'ordre supérieur. D'autre part, la question du décodage est explorée et un nouvel algorithme est proposé, capable de concurrencer le décodeur par liste tout en conservant une complexité moyenne proche de celle du décodeur SC.

Le **Chapitre 1** est revenu sur la structure récursive des codes polaires, décrite par la puissance de Kronecker d'une matrice noyau. Le phénomène de polarisation, selon lequel N réalisations indépendantes du même canal W permettant d'obtenir N canaux virtuels, dont les informations mutuelles tendent asymptotiquement vers 0 ou 1 est détaillé. La notion d'ordre partiel universel des canaux virtuels, conséquence de la structure récursive du graphe polaire, est également abordée. Une caractérisation simple de cet ordre partiel est proposée, réunissant les deux règles connues dans l'état de l'art. Cet ordre partiel induit des propriétés structurelles très particulières des codes polaires impactant le spectre des distances des codes. Une nouvelle formule est proposée, permettant de facilement calculer, pour tout code polaire, le nombre de mots de codes de poids minimaux, ainsi qu'une méthode de calcul par accumulation. Cette nouvelle formule est également exploitée afin de caractériser une nouvelle famille de codes, appelés codes SDO, garantissant, pour tout couple (N, K) la maximisation de la distance minimale en même temps que la minimisation du nombre de mot de codes de poids faible. Le passage en revue des principaux algorithmes de décodage existants pour les codes polaires a mis en évidence qu'un certains nombre d'entre eux peuvent bénéficier d'une construction non plus optimisée pour le décodeur par annulation successive, mais ayant, sinon une meilleure distance minimale, au moins un nombre de mots de code de poids faible réduit. En fournissant un lien explicite simple entre les indices des canaux virtuels et leur impact sur le spectre des distances du code, la nouvelle formule proposée constitue un outil puissant pour optimiser les codes en fonction des algorithmes de décodage.

Le **Chapitre 2** s'est intéressé à la question de la construction de codes polaires flexibles en longueur. Les deux principales méthodes pour cela, à savoir le poinçonnage et le raccourcissement, ont été considérées. Tout d'abord, l'équivalence de deux motifs de poinçonnage est caractérisée à partir de permutations élémentaires appliquées sur les motifs considérés. Pour chaque classe d'équivalence ainsi définie, un unique représentant, appelé motif primitif, est explicité. Une sous-catégorie de motifs primitifs, appelés motifs symétriques, est également définie et une caractérisation précise est fournie à partir des lignes de

la matrice G_N. Il est notamment mis en évidence le fait que les principaux motifs de l'état de l'art sont symétriques. Le dénombrement des motifs de ces deux catégories est abordé. Dans un deuxième temps, le cas du raccourcissement des codes est étudié, mais il est montré qu'il s'agit d'un problème tout à fait connexe à celui du poinçonnage, de sorte que la même classification est introduite. Plus encore, il est mis en évidence le fait qu'une exploration exhaustive des motifs de poinçonnage peut être très simplement transposée aux motifs de raccourcissement. Un algorithme est proposé afin de déterminer le meilleur motif, poinçonnage et raccourcissement compris, pour le décodeur SC et les résultats fournissent une confirmation nette de l'avantage de l'utilisation du poinçonnage pour les rendements faibles et du raccourcissement pour les rendements élevés. Un second algorithme est proposé pour les codes plus longs, pour lesquels une exploration exhaustive n'est plus envisageable, exploitant la caractérisation des motifs symétriques. Finalement, le chapitre s'est efforcé de mettre en évidence un lien entre un code poinçonné ou raccourci, et un code polaire obtenu en combinant plusieurs noyaux. Il est montré que ce lien existe lorsque les noyaux sont eux-mêmes assimilables à des codes poinçonnés ou raccourcis par des motifs symétriques. Dès lors, la notion d'exposant d'erreur, utilisée essentiellement pour estimer la vitesse de la polarisation d'une matrice noyau, est étudiée sur les matrices de transformation de code poinçonnés ou raccourcis. Il est notamment montré que cet exposant se calcule simplement pour les codes raccourcis par un motif symétrique, que cet exposant est maximal notamment pour le motif QUS, et finalement que celui-ci ne dépasse jamais la valeur $\frac{1}{2}$.

Le **Chapitre 3** a abordé la question du décodage des codes polaires en longueur finie, et plus particulièrement le décodage à inversion. Ce décodeur consiste à effectuer des tentatives de décodage en série, tant qu'aucune ne vérifie le CRC ou qu'un nombre maximal de tentatives n'a pas été atteint. Une étude préliminaire de l'impact de corriger un nombre ω d'erreurs est exploitée afin de mettre en évidence l'avantage de parvenir à corriger plusieurs erreurs. Un nouvel algorithme de décodage est proposé, appelé décodeur dynamique à inversion, capable de corriger plusieurs erreurs, grâce à la mise à jour d'une liste d'inversions binaires, ordonnées par probabilités de succès décroissantes. En plus de sa capacité à corriger plusieurs erreurs, un soin particulier est porté sur l'optimisation de la métrique. Il est mis en évidence que le processus séquentiel d'un décodage polaire rend la comparaison de trajectoires de longueurs différentes particulièrement délicate. Une nouvelle métrique est proposée, et utilisant un paramètre de perturbation, dont la valeur est optimisée de manière heuristique, et il est montré la nette supériorité de la métrique proposée sur les métriques de l'état de l'art. Par suite, la question de la réduction de la complexité et de la latence du décodeur proposé est envisagée, à travers un décodeur à inversion simplifié, pour lequel les nœuds de rendement 1 du code n'ont plus besoin d'être décodés, et pour une dégradation très légère des performances. Finalement, il est montré qu'il existe de similitudes entre le décodeur à inversion et le décodeur séquentiel pour les codes polaires, et notamment que la métrique proposée peut également être bénéfique pour le décodage séquentiel, en offrant un gain significatif de performance pour une même quantité de mémoire.

Dans le **Chapitre 4**, la question de la combinaison d'un code polaire binaire avec une modulation d'ordre supérieur est considérée. En particulier, celui-ci s'attèle à comparer la modulation multi-niveaux et la modulation à entrelacement de bits lorsque des codes polaires sont utilisés. Il est mis en évidence le fait que les codes polaires se conjuguent parfaitement avec une approche multi-niveaux, grâce à leur excellente flexibilité et la possibilité de facilement résoudre le problème du choix des rendements sur les différents niveaux binaires pour le décodage par annulation successive. Une méthode de construction est également proposée pour optimiser le choix des rendements lorsqu'un décodeur par liste est utilisé, en s'efforçant d'équilibrer les taux d'erreurs estimés grâce à la borne de l'union, estimée à partir du nombre de mots de code de poids faible obtenu par la formule introduite dans le chapitre 1. Finalement, il est mis

en évidence le fait qu'en modulation multi-level, le processus successif des décodeurs polaires peut être exploité par un décodeur opérant continument sur les différents niveaux binaires, plutôt que d'utiliser des décodeurs séparés. Cette stratégie est appliquée au décodeur par liste et au décodeur dynamique à inversion, et garantit dans les deux cas un gain significatif en termes de performances, au prix d'une augmentation modeste de la complexité.

Le **Chapitre 5** a abordé la question des codes polaires non-binaires. En particulier, un noyau non-binaire de dimension $\ell = 2$ est considéré, permettant un processus de décodage tout à fait similaire à celui des codes binaires. La question du choix de la permutation introduite dans la transformation noyau est abordée. Dans un premier temps, il est montré qu'en se contentant d'un unique noyau, un gain significatif pouvait être obtenu, aussi bien pour un canal à entrée binaire que non-binaire, lorsque la permutation est optimisée dans le groupe symétrique de $\mathcal{A}$, ou plus simplement dans le groupe général linéaire, plutôt que lorsque celle-ci est réduite à la multiplication par un coefficient dans le corps de Galois. Une méthode est ensuite proposée afin d'optimiser chaque bloc de permutation du graphe du code non-binaire spécifiquement, aussi bien pour une permutation sur le groupe symétrique que dans le groupe général linéaire ou le corps de Galois. Les simulations montrent qu'un gain substantiel peut être obtenu en optimisant les coefficients dans le corps de Galois par rapport à un choix aléatoire, mais aussi que la différence entre l'optimisation dans le corps de Galois ou dans le groupe général linéaire reste très limitée. Finalement, il apparait que, au prix d'une complexité supérieure, les codes polaires non-binaires supplantent assez significativement leurs homologues binaires.

Perspectives

A partir des résultats présentés dans cette thèse, plusieurs perspectives peuvent être considérées.

La thèse présente une méthode pour obtenir le code appelé SDO, étant, parmi tous les codes vérifiant l'ordre partiel, celui ayant la distance minimale la plus élevée et le plus faible nombre de mots de code de poids faible. S'il a été montré que ce code peut être intéressant pour le décodeur par liste, il est également susceptible d'améliorer les performances d'une large gamme de décodeurs pour les codes polaires. Parmi ceux-ci, le décodeur par propagation de croyance est probablement l'un de ceux qui peut le plus bénéficier d'un tel code, du fait que celui-ci n'est pas adapté pour une concaténation avec un CRC.

Toujours en lien avec les mots de codes de poids faible, un outil inexploité dans la thèse, mais particulièrement prometteur pour un grand nombre de problématiques liées aux codes polaires, consiste en un algorithme capable d'énumérer (et non plus simplement dénombrer) les mots de code de poids faible. Jusqu'à maintenant, la seule approche appliquée dans l'état de l'art consiste à utiliser un décodeur par liste avec une taille de liste considérable, ce qui est à la fois complexe et sans garantie absolue quant à l'exhaustivité du résultat. En revanche, la démonstration de la formule de dénombrement des mots de codes de poids faible exploitant le formalisme polynomial dans [9] fournit tous les outils pour implémenter un algorithme simple et rapide pour résoudre ce problème. Celui-ci pourrait être utilisé pour observer le spectre des distances d'un système concaténé polaire-CRC, optimiser le polynôme générateur du CRC, optimiser les bits de parités (ou bits figés dynamiques) introduits, notamment, dans les standards de la 5G, analyser le spectre des distances d'un code poinçonné ou raccourci,...

Le Chapitre 3 a mis en évidence que le critère de l'exposant d'erreur a une valeur essentiellement asymptotique, et qu'un exposant d'erreur maximal ne garanti pas pour autant de meilleure performances en longueur finie. Pourtant, il reste le critère privilégié pour décider des matrices noyaux pour les codes polaires basés sur un ou plusieurs noyaux. La pertinence de ce critère doit donc être relativisée, mais il

n'est pas évident de déterminer quels critères pourraient affiner ces choix. L'exposant d'échelle, autre critère dont la signification est asymptotique, reste encore peu considéré. Les résultats de l'état de l'art montrent que les meilleures matrices du point de vue de chacun de ces critères sont distinctes. Dès lors, une question ouverte reste de savoir si l'un de ces critères peut être privilégié, ou bien si d'autres critères sont susceptible de fournir une meilleure discrimination des noyaux en longueur finie.

En lien avec la remarque précédente, la question du décodage d'un noyau de dimension $\ell > 2$ reste encore délicate. Cependant, il est certain que toutes les matrices noyaux ne sont pas équivalentes du point de vue de la possibilité d'implémentation d'un décodage efficace. Dès lors, au delà du critère des performances, les noyaux peuvent également être choisis d'après la simplicité du décodage. Une telle approche a été entamée avec les travaux sur les permutations dans le graphe polaire permettant d'améliorer les performances du SC, sans nécessiter une augmentation de la complexité du décodage. Cependant, il est probable qu'un compromis intéressant entre l'augmentation de la complexité et l'amélioration des performances pourrait être obtenu grâce à un choix judicieux du noyau.

Il existe trois grandes approches pour obtenir de bonnes performances en longueur finie: le décodage par liste, le décodage séquentiel et le décodage à inversion. Chacun de ces algorithmes présente un défaut majeur. Le premier a une complexité et une latence relativement élevées. Le deuxième nécessite une taille de mémoire considérable, sous peine de détérioration excessivement problématique des performances. Enfin, le décodage à inversion dynamique proposé constitue une solution efficace pour un nombre réduit de tentatives, mais la latence maximale augmente considérablement avec le nombre de tentatives. A la fin du chapitre 3, un rapprochement entre les décodeurs à inversion et séquentiel était mis en évidence. Ce rapprochement mériterait d'être approfondi afin, peut-être, de trouver un compromis palliant aux défauts respectifs de ces deux décodeurs.

Une possibilité pour réduire la latence du décodeur à inversion proposé serait de considérer l'utilisation de bits figés dynamiques, susceptibles d'éviter au décodeur de décoder l'ensemble d'une trajectoire pour s'assurer que le décodage est correct ou non. Une telle technique pourrait également faciliter l'identification des erreurs et donc potentiellement apporter une amélioration des performances.

Les codes polaires non-binaires constituent une piste particulièrement prometteuse pour atteindre d'excellentes performances en longueur finie. Néanmoins, la complexité du décodage reste un frein en pratique et des algorithmes de décodages ayant une complexité réduite sont nécessaires. De plus, la thèse s'est concentrée sur une constellation 16-QAM, tandis qu'il a été mis en évidence que l'avantage des codes non-binaires par rapport aux codes binaires augmente avec l'ordre de modulation [39], de sorte qu'un gain encore plus important est attendu pour des ordres supérieurs.

ANNEXES

A Généralités sur les codes polaires

A.1 Ordre partiel (Proposition 1)

Nous avons proposé de formaliser l'ordre partiel pour les codes polaires par la formule suivante:

$$\sum_{k=0}^{m} b_n^{(i)}[k] \geq \sum_{k=0}^{m} b_n^{(j)}[k], \quad \forall m \in \{0,\dots,n-1\} \Rightarrow Z(W_N^{(i)}) \leq Z(W_N^{(j)}) \tag{A.1}$$

Démonstration : L'idée est de construire une séquence $i_1, i_2, \dots, i_T$, avec $i_1 = i$ et $i_T = j$, et telle que pour chaque $t = 1,\dots,T-1$, i_t et i_{t+1} vérifient une des deux règles des équations (1.15) et (1.16). En utilisant cette séquence et les règles de l'état de l'art, on en déduit que:

$$Z(W_N^{(i)}) = Z(W_N^{(i_1)}) \leq Z(W_N^{(i_2)}) \leq \cdots \leq Z(W_N^{(i_T)}) = Z(W_N^{(j)})$$

Soit deux indices $(i,j) \in \{0, N-1\}^2$, $i \neq j$, vérifiant l'équation (A.1), et $k_0 < k_1 < \cdots < k_m$ tels que $b_n^{(i)}[k_\ell] \neq b_n^{(j)}[k_\ell]$. Nécessairement, $b_n^{(i)}[k_0] = 1$ et $b_n^{(j)}[k_0] = 0$. Considérons maintenant les deux cas suivants:

- si $b_n^{(i)}[k_\ell] \geq b_n^{(j)}[k_\ell], \forall \ell \geq 1$, d'après l'équation (1.15), on a $Z(W_N^{(i)}) \leq Z(W_N^{(j)})$.
- sinon, soit k_q le premier indice tel que $b_n^{(i)}[k_q] = 0$ (et donc que $b_n^{(j)}[k_q] = 1$). On définit alors l'indice i_1 obtenu par un "left-swap" dans le développement binaire de i entre $b_n^{(i)}[k_0]$ et de $b_n^{(i)}[k_q]$. D'après l'équation (1.16), il s'ensuit que $Z(W_N^{(i)}) \leq Z(W_N^{(i_1)})$

On applique alors le même procédé sur i_1 autant de fois que nécessaire afin de montrer finalement que :

$$Z(W_N^{(i)}) = Z(W_N^{(i_1)}) \leq Z(W_N^{(i_2)}) \leq \cdots \leq Z(W_N^{(i_T)}) = Z(W_N^{(j)})$$

□

A.2 Dénombrement des mots de codes de poids faible

A.2.1 Démonstration de la formule directe (Proposition 2)

Cette partie fait intervenir le formalisme polynomial développé pour les codes RM et appliqué aux codes polaires dans [9]. Nous ne reviendrons pas sur la définition de ce formalisme, le lecteur pouvant se référer par exemple à [81, ch. 6]. Avant d'attaquer la démonstration qui nous occupe, il convient de rappeler le lien entre les lignes de G_N et leurs monomiaux associés donné dans [9, partie II][1]:

$$G_N[i] = \mathrm{ev}(x_0^{1-b_n^{(i)}[n-1]} x_1^{1-b_n^{(i)}[n-2]} \cdots x_{n-1}^{1-b_n^{(i)}[0]}) \tag{A.2}$$

[1] Il y a une légère différence avec [9] du fait que la matrice G_N y est définie de sorte à être triangulaire supérieure. Cela revient à faire un changement d'indice de i à $N-1-i$ et de changer $b_n^{(i)}[k]$ en $b_n^{(N-1-i)}[k] = 1 - b_n^{(i)}[k]$. Il y a également une inversion MSB et LSB pour le développement binaire.

où $ev(g)$ est appelée la fonction d'évaluation du polynôme g et sert à associer à un polynôme un vecteur binaire caractérisé par l'évaluation du polynôme pour toutes les entrées $(u_0, u_1, \ldots, u_{n-1}) \in \{0,1\}^n$. Une conséquence immédiate de cette propriété est la relation entre le degré d'un monomial et le poids de la ligne de G_N associée:

$$deg(i) = \sum_{k=0}^{n-1} (1 - b_n^{(i)}[k]) = n - |B_n^{(i)}| \tag{A.3}$$

La formule donnée dans [9] pour dénombrer les mots de codes de poids minimal fait intervenir la dimension du diagramme de Ferrer des monomiaux et se présente sous la forme:

$$\mu_{min} = 2^{r_+} \sum_{g \in I_{r_+}} 2^{|\lambda_g|}, \tag{A.4}$$

où:

- $r_+ = \min\{\, r \mid \mathcal{C}(\mathcal{I}) \subseteq \mathcal{R}(r,n)\}$. Il s'agit de l'ordre du code RM ayant la même distance minimale que le code considéré. D'où $r_+ = n - d$ avec les notations utilisées dans le manuscrit, où la distance minimale du code polaire est donnée par $\mathrm{D}_{\min} = 2^d$.
- $I_{r_+} = \{f \in \mathcal{I} \mid deg(f) = r_+\}$ est l'ensemble des monomiaux de degré $r_+ = n - d$ associés aux indices $f \in \mathcal{I}$. En utilisant l'équation (A.3), cette expression devient:

$$I_{r_+} = \{i \in \mathcal{I} \mid |B_n^{(i)}| = d\} \tag{A.5}$$

- λ_g est le diagramme de Ferrer du monomial g, et $|\lambda_g|$ représente sa dimension.

Dans [9, prop.10], il est indiqué que si $g = x_{i_1} x_{i_2} \ldots x_{i_{d'}}$ est un monomial de degré d', alors la partition associée à g est donné par $\lambda_g = (i_{d'} - (d'-1), i_{d'-1} - (d'-2), \ldots, i_1 - 0)$. D'où l'on déduit:

$$|\lambda_g| = \sum_{k=1}^{d'} (i_k - (k-1)) \tag{A.6}$$

$$= \sum_{k=1}^{d'} i_k - d'(d'-1)/2 \tag{A.7}$$

En appliquant cette expression aux lignes de poids minimal, on obtient:

$$\begin{aligned}
|\lambda_g| &= \sum_{k=0}^{n-1} k \cdot (1 - b_n^{(i)}[n-1-k]) - \frac{(n-d)(n-d-1)}{2} \\
&= \frac{n(n-1)}{2} - \frac{(n-d)(n-d-1)}{2} - \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[n-1-k] \\
&= \frac{n(n-1)}{2} - \frac{(n-d)(n-d-1)}{2} - \sum_{k=0}^{n-1} (n-1-k) \cdot b_n^{(i)}[k] \\
&= \frac{n(n-1)}{2} - \frac{(n-d)(n-d-1)}{2} - (n-1)d + \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k] \\
&= -\frac{d(d-1)}{2} + \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k]
\end{aligned}$$

Finalement, le nombre de mots de code de poids minimal est donné par:

$$\mu_{min} = 2^{n-d} \cdot 2^{-d(d-1)/2} \sum_{\substack{i \in \mathcal{I} \\ |B_n^{(i)}| = d}}^{n-1} 2^{s_n(i)}, \tag{A.8}$$

avec $s_n(i)$ défini par:

$$s_n(i) = \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k] \tag{A.9}$$

A.2.2 Démonstration de la méthode par accumulation (Proposition 3)

Le fait que les codes composites sont eux-mêmes CMD découle directement de [43, prop.1]. Pour montrer que $D_1 \geq D_2$, on procède par l'absurde en supposant que $\mathcal{I}_1 \not\subset \mathcal{I}_2$. Dès lors, il existe $i \in \{0, N/2-1\}$ tel que $i \in \mathcal{I}_1$ et $i \notin \mathcal{I}_2$. Or, on a $B_n^{(i)} = (0, b_{n-1}^{(i)}[1], \ldots, b_{n-1}^{(i)}[n-1])$ et $B_n^{(i+N/2)} = (1, b_{n-1}^{(i)}[1], \ldots, b_{n-1}^{(i)}[n-1])$, ce qui contredit la condition d'ordre partiel. Finalement, si $\mathcal{I}_1 \subset \mathcal{I}_2$, alors $D_1 \geq D_2$.

Démontrons maintenant la proposition 2. La détermination de la distance minimale suit la règle bien connue [93, chap.3]:

$$\mathrm{D}_{\min} = \min(D_1, 2D_2) \tag{A.10}$$

Commençons par expliciter le lien entre $s_n(i)$ et $s_{n-1}(i)$:

$$\begin{aligned}
s_n(i) &= \sum_{j=0}^{n-1} j \cdot b_n^{(i)}[j] \\
&= \sum_{j=1}^{n-1} j \cdot b_n^{(i)}[j] \\
&= \sum_{j=0}^{n-2} (j+1) \cdot b_n^{(i)}[j+1] \\
&= \begin{cases} \sum_{j=0}^{n-2} (j+1) \cdot b_{n-1}^{(i)}[j] & \text{si } i < N/2 \\ \sum_{j=0}^{n-2} (j+1) \cdot b_{n-1}^{(i-N/2)}[j] & \text{si } i \geq N/2 \end{cases} \\
&= \begin{cases} s_{n-1}^{(i)} + |B_{n-1}^{(i)}| & \text{si } i < N/2 \\ s_{n-1}^{(i-N/2)} + |B_{n-1}^{(i-N/2)}| & \text{si } i \geq N/2 \end{cases}
\end{aligned}$$

On considère tout d'abord le cas où $\mathrm{D}_{\min} = D_1 = 2D_2$:

$$
\begin{aligned}
\mu_{\min} &= 2^{n-\frac{d(d+1)}{2}} \sum_{\substack{i\in\mathcal{I} \\ |B_n^{(i)}|=d}} 2^{s_n(i)} \\
&= 2^{n-\frac{d(d+1)}{2}} \left(\sum_{\substack{i\in\mathcal{I}_1 \\ |B_n^{(i)}|=d}} 2^{s_n(i)} + \sum_{\substack{i\in\mathcal{I}_2 \\ |B_n^{(i+N/2)}|=d}} 2^{s_n(i+N/2)} \right) \\
&= 2^{n-\frac{d(d+1)}{2}} \left(\sum_{\substack{i\in\mathcal{I}_1 \\ |B_{n-1}^{(i)}|=d}} 2^{s_{n-1}(i)+|B_{n-1}^{(i)}|} + \sum_{\substack{i\in\mathcal{I}_2 \\ |B_{n-1}^{(i)}|=d-1}} 2^{s_{n-1}(i)+|B_{n-1}^{(i)}|} \right) \\
&= 2^{n-1-\frac{d(d+1)}{2}} \cdot 2^{d+1} \left(\sum_{\substack{i\in\mathcal{I}_1 \\ |B_{n-1}^{(i)}|=d}} 2^{s_{n-1}(i)} \right) + 2^{n-1-\frac{d(d-1)}{2}} \left(\sum_{\substack{i\in\mathcal{I}_2 \\ |B_{n-1}^{(i)}|=d-1}} 2^{s_{n-1}(i)} \right) \\
&= 2 \cdot D_1 \cdot \mu_1 + \mu_2
\end{aligned}
$$

Cette dernière ligne utilisant le fait que les codes composites sont eux-mêmes des CMD, dont les multiplicités se calculent donc par (1.19). Pour démontrer les cas $D_1 = D_2$ et $D_1 > 2D_2$ de la Proposition 3, il suffit de considérer la démonstration ci-dessus lorsque $\{i \in \mathcal{I}_1, |B_{n-1}^{(i)}| = d\} = \emptyset$ et $\{i \in \mathcal{I}_2, |B_{n-1}^{(i)}| = d-1\} = \emptyset$ respectivement.

Il s'avère qu'en réalité le cas $D_1 = D_2$ ne survient que dans un cas très particulier:

Proposition 24 *Si $\mathcal{C}_1(N/2, K_1, \mathcal{I}_1)$ et $\mathcal{C}_2(N/2, K_2, \mathcal{I}_2)$ sont les codes composites d'un code monomial $\mathcal{C}(N, K, \mathcal{I})$, alors nécessairement:*

$$D_1 = D_2 \Rightarrow \mathcal{I}_1 = \mathcal{I}_2 = \emptyset \;\; ou \;\; \mathcal{I}_1 = \mathcal{I}_2 = \{0, \dots, N/2-1\} \tag{A.11}$$

Démonstration:
Soit le code $\mathcal{C}_1(N/2, K_1, \mathcal{I}_1)$ avec $K_1 \in \{1, N/2-1\}$ et $D_1 = 2^{d_1}$. Dès lors (puisque $|\mathcal{I}_1| = K_1 > 0$):

$$\exists i_1 \in \mathcal{I}_1, \text{tel que } |B_{n-1}^{(i_1)}| = d_1 \tag{A.12}$$

De plus, puisque $K_1 < N/2 - 1$, il est clair que $0 \notin \mathcal{I}_1$. D'où $i_1 > 0$ et:

$$\exists k \in \{0, \dots, n-1\}, \text{tel que } b_{n-1}^{(i_1)}[k] = 1 \tag{A.13}$$

Considérons l'index $i_2 = i_1 - 2^{n-1-k}$. Montrons que, d'après les règles d'ordre partiel, $i_2 \in \mathcal{I}_2$:

$$
\begin{aligned}
B_n^{(i_1)} &= (0, b_{n-1}^{(i_1)}[0], \dots, b_{n-1}^{(i_1)}[k], \dots, b_{n-1}^{(i_1)}[n-1]) \\
B_n^{(i_2+N/2)} &= (1, b_{n-1}^{(i_1)}[0], \dots, \quad 0 \quad, \dots, b_{n-1}^{(i_1)}[n-1])
\end{aligned}
$$

La somme cumulée du vecteur $B_n^{(i_2+N/2)}$ est toujours supérieure à $B_n^{(i_1)}$ d'où nécessairement $i_2 \in \mathcal{I}_2$. Finalement, la distance minimale de $\mathcal{C}_2$ est telle que:

$$D_2 \leq 2^{|B_{n-1}^{(i_2)}|} = 2^{d_1-1} < D_1 \tag{A.14}$$

La conséquence de cette propriété est que le cas $D_1 = D_2$ ne survient en réalité que si les codes composites sont tous deux constitués soit uniquement de bits d'information soit uniquement de bits figés.

A.2.3 Maximisation de la multiplicité (Proposition 4)

Le point délicat de la démonstration est de prouver que le code décrit est bien CMD. Le fait qu'il maximise la distance minimale et minimise la multiplicité découle ensuite simplement de l'équation (1.19).

Il a été indiqué dans la section 1.2.2 que si i et j sont tels que $|B_n^{(i)}| > |B_n^{(j)}|$, alors $j \not\succeq i$. Dès lors le fait de sélectionner les lignes par poids décroissants ne peut contredire l'ordre partiel. Il reste à montrer que si deux lignes sont de même poids, alors les sélectionner par valeurs de s_n croissantes ne contredit pas l'ordre partiel. Pour cela, il suffit de montrer la propriété suivante:

Proposition 25 *$\forall (i,j) \in \{0,\dots,N-1\}^2$, $i \neq j$ avec $|B_n^{(i)}| = |B_n^{(j)}|$, alors :*

$$i \succeq j \Rightarrow s_n(i) < s_n(j) \tag{A.15}$$

Démonstration: Récrivons tout d'abord le calcul de $s_n(i)$ de la manière suivante:

$$\begin{aligned} s_n(i) &= \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k] \\ &= \sum_{k=0}^{n-1} \sum_{k'=k}^{n-1} b_n^{(i)}[k'] \\ &= \sum_{k=0}^{n-1} \left(\sum_{k'=0}^{n-1} b_n^{(i)}[k'] - \sum_{k'=0}^{k-1} b_n^{(i)}[k'] \right) \\ &= n|B_n^{(i)}| - \sum_{k=0}^{n-1} \sum_{k'=0}^{k-1} b_n^{(i)}[k'] \end{aligned}$$

D'où l'on obtient finalement, en supposant que $i \succeq j$:

$$s_n(i) - s_n(j) = \underbrace{n|B_n^{(i)}| - n|B_n^{(j)}|}_{=0} + \sum_{k=0}^{n-1} \underbrace{\left(\sum_{k'=0}^{k-1} b_n^{(j)}[k'] - \sum_{k'=0}^{k-1} b_n^{(i)}[k'] \right)}_{\leq 0}$$

Puisque $i \neq j$, on obtient finalement que $s_n(i) < s_n(j)$.

La conséquence est que si deux indices i et j sont tels que $|B_n^{(i)}| = |B_n^{(j)}|$ et que $s_n(i) = s_n(j)$, alors peu importe le critère de discrimination entre les deux, le code restera CMD et minimisera toujours la multiplicité.

A.2.4 Métrique de caractérisation d'un code SDO (Proposition 5)

Tout d'abord, notons que:

$$s_n(i) = \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k] \leq \sum_{k=0}^{n-1} k = \frac{n(n-1)}{2}$$

Soit $\nu \geq \frac{n(n-1)}{2}$. La métrique associée à l'index i peut être définie par:

$$\begin{aligned} \Gamma'(i) &= |B_n^{(i)}| \cdot \nu - s_n(i) \\ &= \nu \sum_{k=0}^{n-1} b_n^{(i)}[k] - \sum_{k=0}^{n-1} k \cdot b_n^{(i)}[k] \\ &= \nu \left(\sum_{k=0}^{n-1} b_n^{(i)}[k] \left(1 - \frac{k}{\nu} \right) \right) \end{aligned}$$

B Codes polaires flexibles

B.1 Motif d'effacement (Proposition 7)

On démontre ici la Proposition 7, à savoir que tout motif d'effacement est une réunion de lignes de la matrice G_N. Dans [5], Arikan fournit une formule simple pour caractériser les coefficients de la matrice G_N par:

$$G_N^{(i)}[j] = \prod_{k=0}^{n-1} (1 \oplus b_n^{(i)}[k] \oplus b_n^{(i)}[k] \cdot b_n^{(j)}[k]), \forall (i,j) \in \{0,\dots,N-1\} \tag{B.1}$$

Nous fournissons une autre formule tout aussi simple et équivalente:

$$G_N^{(i)}[j] = \prod_{k=0}^{n-1} b_n^{(i)}[k]^{b_n^{(j)}[k]}, \forall (i,j) \in \{0,\dots,N-1\} \tag{B.2}$$

L'égalité entre les deux formules provient du fait que $a^b = ab \oplus b \oplus 1$, dont la table de vérité est donnée dans le tableau B.1. Cette formule sera utilisée dans la démonstration de la Proposition 7 ci-dessous.

Démonstration : Soit P un motif de poinçonnage et $E[P]$ le motif d'effacement associé. Soit $i \in \{0,\dots,n-1\}$ tel que $E[P](i) = 1$. Rappelons qu'un code polaire poinçonné vérifie la propriété d'ordre partiel donnée par l'équation (1.15), d'où:

$$\forall k \in \{0,\dots,n-1\}, b_n^{(i)}[k] \geq b_n^{(j)}[k] \Rightarrow Z(W[P]_N^{(i)}) \leq Z(W[P]_N^{(j)}) \tag{B.3}$$

Dès lors, le fait que $E[P](i) = 1$ impose que $E[P](j) = 1$ pour tout $j \in \{0,\dots,N-1\}$ tel que:

$$b_n^{(i)}[k] \geq b_n^{(j)}[k] \ \forall k \in \{0,\dots,n-1\} \tag{B.4}$$

Or:

$$\begin{aligned} b_n^{(i)}[k] \geq b_n^{(j)}[k] \ \forall k \in \{0,\dots,n-1\} &\Rightarrow \prod_{k=0}^{n-1} b_n^{(i)}[k]^{b_n^{(j)}[k]} = 1 \\ &\Rightarrow G_N^{(i)}[j] = 1 \end{aligned}$$

Table B.1: *Table de vérité de* $ab \oplus b \oplus 1$ *et* a^b

a	b	$ab \oplus b \oplus 1$	a^b
0	0	1	1
0	1	0	0
1	0	1	1
1	1	1	1

Autrement dit, si $E[P](i) = 1$ alors $E[P](j) = 1$ pour tout indice j tel que $G_N^{(i)}[j] = 1$, et donc $G_N^{(i)} \subseteq E[P]$. Réitérer le même raisonnement pour tout indice i tel que $E[P](i) = 1$ prouve le résultat. □

B.2 Motif primitif (Proposition 10)

Il s'agit de démontrer la Proposition 10, selon laquelle le motif primitif $\Phi(P)$ vérifie:

(i) $\Phi(P) \approx P$

(ii) $P \approx P' \Leftrightarrow \Phi(P) = \Phi(P')$

Démonstration : (Proposition 10) Les démonstrations de (i) et (ii) s'obtiennent facilement en exploitant la représentation des motifs par un arbre binaire. Nécessairement, l'arbre de $\Phi(P)$ est isomorphe à l'arbre de P, et les motifs sont donc équivalents. De plus, pour tout motif P, il existe un unique arbre monotone isomorphe à celui de P, ce qui prouve que $P \approx P' \Rightarrow \Phi(P) = \Phi(P')$. Le sens inverse s'obtient par le fait que si deux motifs P et P' sont tels que les arbres binaires associés à $\Phi(P)$ et $\Phi(P')$ sont identiques, alors les arbres associés à P et P' sont isomorphes et $P \approx P'$. □

Démontrons maintenant la Proposition 11, indiquant comment construire récursivement les motifs primitifs.

Démonstration : (Proposition 11) Il convient de remarquer que tout arbre monotone (associé à un motif P) se décompose en deux arbres monotones dont la valeur des nœuds racines, notées $N_p^{(1)}$ et $N_p^{(2)}$ respectivement, vérifie $N/2 \geq N_p^{(1)} \geq N_p^{(2)}$ avec $N_p^{(1)} + N_p^{(2)} = N_p$. A chacun de ces deux arbres monotones peut être associé un motif de poinçonnage, de longueur $N/2$, noté P_1 et P_2 respectivement. Ceux-ci sont liés au motif P par la relation:

$$B_N(P) = [B_{N/2}(P_1)\ B_{N/2}(P_2)], \tag{B.5}$$

ce qui prouve le résultat. □

B.3 Motif symétrique

La démonstration de la prochaine propriété utilisera une récurrence basée sur la structure récursive du graphe polaire. On se propose d'introduire les notations et de démontrer quelques propriétés préliminaires.

D'après la figure B.1, un motif de poinçonnage P de longueur N donne, à l'issue du premier étage de noyau en deux motifs P_+ et P_- de longueur $\frac{N}{2}$. Si l'on note $P = [P_a\ P_b]$ où P_a et P_b correspondent aux indices $i \in \{0, \ldots, \frac{N}{2} - 1\}$ et $i \in \{\frac{N}{2}, \ldots, N-1\}$ respectivement, alors:

$$P_+ = \mathbf{1}_N - (\mathbf{1}_N - P_a) * (\mathbf{1}_N - P_b)$$
$$P_- = P_a * P_b$$

où $*$ représente le produit terme à terme. Les deux motifs P_+ et P_- sont les motifs de poinçonnages des sous-blocs polaires de dimension $\frac{N}{2}$, et génèrent donc deux motifs d'effacements E_+ et E_- respectivement. Le motif d'effacement $E(P)$ est donné par la concaténation des deux, *i.e.* $E[P] = [E_+\ E_-]$.

Proposition 26 *Pour tout motif $P \in \{0,1\}^N$, les propriétés suivantes sont vérifiées:*

(i) $P_- \subseteq P_+$

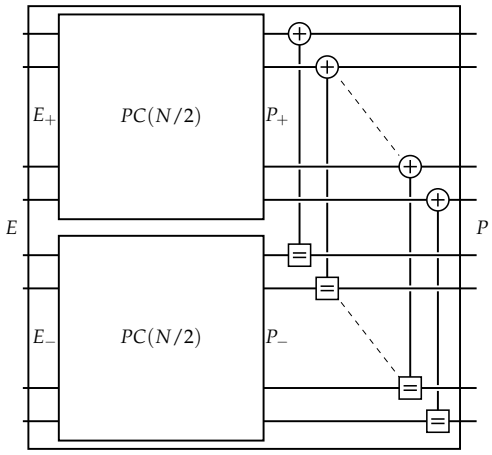


FIGURE B.1: *Construction récursive du graphe polaire*

(ii) $E_- \subseteq E_+$

Démonstration : Pour (i), il suffit de montrer que $P_+(i) \geq P_-(i), \forall i \in \{0,\dots,\frac{N}{2}\}$. Cela suit directement le fait que $1-(1-a)\cdot(1-b) = a+b-ab \geq ab$. La propriété (ii) est la conséquence de (i). Puisque $P_- \subseteq P_+$, alors $Z(W[P_+]_{\frac{N}{2}}^{(i)}) \geq Z(W[P_-]_{\frac{N}{2}}^{(i)})$, d'où $Z(W[P_-]_{\frac{N}{2}}^{(i)}) = 1 \Rightarrow Z(W[P_+]_{\frac{N}{2}}^{(i)}) = 1$, soit $E_- \subseteq E_+$. □

Proposition 27 *P est une réunion de lignes de G_N si et seulement si P_+ et $P_-]$ sont des réunions de lignes de $G_{N/2}$ et $P_- \subseteq P_+$. On a de plus $P = [P_+\ P_-]$.*

Démonstration : Commençons par l'implication directe. Supposons que $P = \bigvee_{i\in\mathcal{L}} G_N^{(i)}$. Puisque les lignes de G_N sont telles que $G_N = [G_{N/2}\ G_{N/2}]$ ou $G_N = [G_{N/2}\ \mathbf{0}_{N/2}]$, on a $P(i) \geq P(i+N/2), \forall i \in \{0,\dots,N/2-1\}$. D'où:

$$\begin{aligned}
P_-(i) &= P(i)\cdot P(i+\frac{N}{2}) \\
&= P(i+\frac{N}{2}) \\
P_+(i) &= 1-(1-P(i))\cdot(1-P(i+\frac{N}{2})) \\
&= P(i)+P(i+\frac{N}{2})+P(i+\frac{N}{2})\cdot P(i) \\
&= P(i)
\end{aligned}$$

D'où:

$$\begin{aligned}
P_- &= \bigvee_{\substack{i\in\mathcal{L} \\ i\geq N/2}} G_{N/2}^{(i-N/2)} \\
P_+ &= \bigvee_{\substack{i\in\mathcal{L} \\ i<N/2}} G_{N/2}^{(i)}
\end{aligned}$$

Ce qui prouve le sens direct, de même que $P = [P_+ \; P_-]$. Pour démontrer le sens inverse, on suppose que $P_- = \bigvee_{i \in \mathcal{L}_-} G^{(i)}_{N/2}$ et $P_+ = \bigvee_{i \in \mathcal{L}_+} G^{(i)}_{N/2}$. En notant $P = [P_a \; P_b]$, et puisque $P_- \subseteq P_+$, on a:

$$\begin{aligned} P_a &= P_+ \\ P_b &= 1 - (1 - P_+) * (1 - P_-) \\ &= P_- \end{aligned}$$

D'où finalement:

$$P = [P_+ \; P_-] = \bigvee_{i \in \mathcal{L}_-} G^{(i+N/2)}_N \bigvee_{i \in \mathcal{L}_+} G^{(i)}_N \tag{B.6}$$

□

Le motif P peut s'exprimer comme la concaténation de deux motifs de longueur $\frac{N}{2}$, étant tous deux une réunion de lignes de $G_{\frac{N}{2}}$, à partir de:

$$\begin{aligned} \mathcal{L}_+ &= \{i(mod\frac{N}{2}), i \in \mathcal{L}\} \\ \mathcal{L}_- &= \{i(mod\frac{N}{2}), i \in \mathcal{L}, i \geq \frac{N}{2}\} \end{aligned}$$

Dès lors, on a $P = [P_+ P_-]$ avec $P_+ = \bigvee_{i \in \mathcal{L}_+} G^{(i)}_N$ et $P_- = \bigvee_{i \in \mathcal{L}_-} G^{(i)}_N$. D'après l'hypothèse de récurrence, $E[P_+] = P_+$ et $E[P_-] = P_-$. Notons que $\mathcal{L}_- \subseteq \mathcal{L}_+$, d'où $P_- \subseteq P_+$ et:

$$P(i) = \begin{cases} P_+(i) & \text{si } i < \frac{N}{2} \\ P_-(i) & \text{si } i \geq \frac{N}{2} \end{cases} \tag{B.7}$$

On peut maintenant démontrer la Proposition 12 :

$$E[P] = P \Leftrightarrow P = \mathbf{0}_N \text{ ou } \exists \mathcal{L} \subset \{0, \ldots, N-1\}, P = \bigvee_{i \in \mathcal{L}} G^{(i)}_N \tag{B.8}$$

Démonstration : (Proposition 12) D'après la proposition 7, on a soit $E[P] = \mathbf{0}_N$ soit $E[P]$ est une réunion de lignes de G_N, ce qui prouve immédiatement le sens direct. La démonstration du sens inverse se fait par récurrence. L'initialisation au niveau du noyau provient du fait que les seuls motifs d'effacement possible sont $[0,0]$, $[1,0]$ et $[1,1]$. La propriété précédente prouve que $P = [P_+ P_-]$, P_+ et P_- étant eux-mêmes des réunion de lignes de $G_{\frac{N}{2}}$. De fait, d'après l'hypothèse de récurrence:

$$E[P] = [E[P_+] \; E[P_-]] = [P_+ \; P_-] = P$$

□

Démontrons maintenant la Proposition 13, à savoir que tout motif symétrique est le motif primitif de sa classe d'équivalence.

Démonstration : (Proposition 13) La démonstration de cette propriété repose sur le fait que toute ligne de G_N est invariante par la fonction ϕ. Pour cela rappelons que ϕ applique des permutations élémentaires $\phi_{k,j}$ en permutant les vecteurs P_a et P_b lorsque $P_a <_{lex} P_b$, où:

$$\begin{aligned} P_a &= \{P(k + q \cdot 2^{n-j}), q \in \{0, \ldots 2^{j-1}\}\} \\ P_b &= \{P(k + q \cdot 2^{n-j} + 2^{n-1-j}), q \in \{0, \ldots 2^{j-1}\}\} \end{aligned}$$

avec $j \in \{0,\dots,n-1\}$ et $k \in \{0,\dots,2^{n-1-j}-1\}$. Il suffit de montrer que si P est une ligne de G_N, alors $P_a \geq_{lex} P_b$:

$$G_N^{(i)}[k+q\cdot 2^{n-j}] = \prod_{\ell=0}^{n-1} b_n^{(i)}[\ell]^{b_n^{(k+q\cdot 2^{n-j})}[\ell]}$$
$$G_N^{(i)}[k+q\cdot 2^{n-j}+2^{n-1-j}] = \prod_{\ell=0}^{n-1} b_n^{(i)}[\ell]^{b_n^{(k+q\cdot 2^{n-j}+2^{n-1-j})}[\ell]}$$

Notons que $b_n^{(k+q\cdot 2^{n-j})}[\ell] \leq b_n^{(k+q\cdot 2^{n-j}+2^{n-1-j})}[\ell], \forall \ell \in \{0,\dots,n-1\}$. D'où:

$$G_N^{(i)}[k+q\cdot 2^{n-j}] \geq G_N^{(i)}[k+q\cdot 2^{n-j}+2^{n-1-j}]$$

Ce qui prouve bien que toute ligne de G_N est invariante par $\phi_{k,j}$ et donc par ϕ. Finalement:

$$\Phi(P) = \phi(\bigvee_{i\in\mathcal{L}} G_N^{(i)}) = \bigvee_{i\in\mathcal{L}} \phi(G_N^{(i)}) = \bigvee_{i\in\mathcal{L}} G_N^{(i)} = P$$

□

Démontrons maintenant que la permutation à inversion de bit d'une ligne de la matrice G_N reste une ligne de G_N.

Démonstration : (Proposition 14)

$$\begin{aligned} B_N(G_N^{(i)})[j] &= G_N^{(i)}[B_N(j)]) \\ &= \prod_{k=0}^{n-1} b_n^{(i)}[k]^{\pi_{lr}(b_n^{(j)}[k])} \\ &= \prod_{k=0}^{n-1} \pi_{lr}(b_n^{(i)})[k]^{b_n^{(j)}[k]} \\ &= G_N^{(\ell)}[j]) \end{aligned}$$

avec m tel que $B_n^{(m)} = \pi_{lr}(B_n^{(i)})$. □

Finalement, démontrons que les motifs symétriques peuvent s'obtenir par récurrence.

Démonstration : (Proposition 15) Soit P_0 et P_1 deux motifs symétriques de longueurs $\frac{N}{2}$ tels que $P1 \subseteq P_0$. Dès lors, il existe $\mathcal{L}_0$ et $\mathcal{L}_1$ avec $\mathcal{L}_1 \subseteq \mathcal{L}_0$ tels que :

$$P_0 = \bigvee_{i\in\mathcal{L}_0} G_{N/2}^{(i)} \text{ et } P_1 = \bigvee_{i\in\mathcal{L}_1} G_{N/2}^{(i)}$$

Soit $P = [P_0\ P_1]$. P peut s'obtenir par:

$$P = \left(\bigvee_{i\in\mathcal{L}_0} G_N^{(i)}\right) \vee \left(\bigvee_{i\in\mathcal{L}_1} G_N^{(i+N/2)}\right)$$

ce qui prouve que P est bien symétrique. Le sens inverse se démontre en partant du motif P symétrique et en montrant que celui-ci se décompose en deux motifs symétriques P_0 et P_1 en exploitant le fait que les lignes de G_N s'écrivent $G_N^{(i)} = [G_{N/2}^{(i)}\ \mathbf{0}_{N/2}]$ si $i < N/2$ et $G_N^{(i)} = [G_{N/2}^{(i)}\ G_{N/2}^{(i)}]$ sinon. □

B.3.1 Motifs de l'état de l'art

On veut démontrer que le motif de poinçonnage P obtenu en sélectionnant les N_p moins bons canaux virtuels du code mère est un motif symétrique, *i.e.* une réunion de lignes de G_N. Cette démonstration est semblable à celle prouvant que tout motif d'effacement $E[P]$ est une réunion de lignes de G_N. Soit i tel que $P(i) = 1$. Dès lors, puisque le code mère vérifie l'ordre partiel donné par l'équation (1.15), on est sûr que $G_N^{(i)} \subseteq P$. Réitérer le même raisonnement pour tout i tel que $P(i) = 1$ donne le résultat.

Le même raisonnement s'applique au cas du raccourcissement en montrant que si i est tel que $S(i) = 1$, alors $\pi_{lr}(G_N^{(N-1-i)}) \subseteq S$.

B.4 Énumération des motifs de poinçonnage et de raccourcissement (Proposition 17)

Démontrons que l'ordre du motif QUP est donné par $\lambda_{QUP} = \sum_{k=0}^{n} b_n^{(N_p)}[k] \leq n$.

Démonstration : (Proposition 17) L'ordre du motif QUP de longueur $N = 2^n$ et de poids N_p, noté $\lambda_n(N_p)$, peut se calculer récursivement de la manière suivante:

$$\lambda_n(N_p) = \begin{cases} \lambda_{n-1}(N_p) & \text{si } N_p < N/2 \\ \lambda_{n-1}(N_p - \frac{N}{2}) + 1 & \text{sinon.} \end{cases} \tag{B.9}$$

Cette récurrence est due au fait que si $N_p \geq N/2$, alors $G_N^{(N/2-1)} \subseteq P$ puisque $G_N^{(N/2-1)} = [\mathbf{1}_{N/2}\, \mathbf{0}_{N/2}]$. On considère alors le motif QUP de longueur $N/2$ et de poids $N_p' = N_p - N/2$ et on réitère la même opération jusqu'à ce que $N_p = 0$. Il apparaît clairement que cela revient à faire la décomposition binaire de N_p et à calculer la somme, ce qui est le résultat annoncé. □

B.5 Poinçonnage, Raccourcissement et structures multi-noyaux

B.5.1 Combinaison de noyaux (Proposition 18)

On se propose de démontrer la Proposition 18 sur permettant d'obtenir le motif de poinçonnage correspondant à la combinaison de deux noyaux poinçonnés par des motifs symétriques. Pour cela, plusieurs résultats préliminaires sont nécessaires.

Proposition 28

$$(A \vee B) \otimes C = (A \otimes C) \vee (B \otimes C), \tag{B.10}$$

Démonstration : Notons que:

$$A \vee B = A + B - A * B$$

ou $A * B$ dénote la multiplication élément par élément (produit de Hadamard). Dès lors:

$$\begin{aligned}(A \vee B) \otimes C &= (A + B - A * B) \otimes C \\ &= A \otimes C + B \otimes C - (A * B) \otimes C \\ &= A \otimes C + B \otimes C - (A \otimes C) * (B \otimes C) \\ &= (A \otimes C) V (B \otimes C)\end{aligned}$$

□

Proposition 29 *Le produit de kronecker d'une ligne de G_{N_1} et de G_{N_2} est une ligne de $G_{N_1 \cdot N_2}$*

Démonstration : Puisque $G_{N_1 \cdot N_2} = G_{N_1} \otimes G_{N_2}$, il s'ensuit que $G_{N_1}^{(i)} \otimes G_{N_2}^{(j)}$ correspond à la ligne $i \cdot N_2 + j$ de $G_{N_1 \cdot N_2}$. □

Nous allons maintenant démontrer la Proposition 18.

Démonstration : (Proposition 18)

Les deux résultats ci-dessus permettent de démontrer que le motif P est bien symétrique. En effet:

$$\begin{aligned}P &= P_1 \otimes \mathbf{1}_{N_2} + \mathbf{1}_{N_1} \otimes P_2 - P_1 \otimes P_2 \\ &= (P_1 \otimes \mathbf{1}_{N_2}) \vee (\mathbf{1}_{N_1} \otimes P_2)\end{aligned}$$

Puisque P_1 est une réunion de lignes de G_{N_1} et $\mathbf{1}_{N_2}$ une ligne de G_{N_2}, alors $P_1 \otimes \mathbf{1}_{N_2}$ est une réunion de lignes de $G_{N_1 \cdot N_2}$. Le même raisonnement s'applique à $\mathbf{1}_{N_1} \otimes P_2$. Finalement, P est donc une réunion de lignes de $G_{N_1 \cdot N_2}$ et est donc symétrique.

Il s'agit maintenant de s'assurer que la matrice $G[P]_{N_1 \cdot N_2} = G[P_1]_{N_1} \otimes G[P_2]_{N_2}$ lorsque $P = (P_1 \otimes \mathbf{1}_{N_2}) \vee (\mathbf{1}_{N_1} \otimes P_2)$. Tout d'abord, appelons $P_2' = (\mathbf{1}_{N_1} \otimes P_2) = [P_2\ P_2\ \dots\ P_2]$. D'après la définition du produit de Kronecker, il est clair que $G[P_2']_{N_1 \cdot N_2} = G_{N_1} \otimes G[P_2]_{N_2}$. En effet, la matrice $G[P_2']_{N_2}$ est obtenue en supprimant les colonnes et les lignes d'indices i tels que $P_2'(i) = 1$, mais cela revient à remplacer chaque matrice G_{N_2} par $G[P_2]_{N_2}$. Un raisonnement similaire peut être effectué en considérant le motif $P_1' = (P_1 \otimes \mathbf{1}_{N_2})$, pour lequel on a $G[P_1']_{N_1 \cdot N_2} = G[P_1]_{N_1} \otimes G_{N_2}$. Finalement, $G[P_1 \vee P_2]_{N_1 \cdot N_2} = G[P_1]_{N_1} \otimes G[P_2]_{N_2}$. □

B.5.2 Exposant d'erreur des matrices QUP

Démonstration : (Proposition 20)

On veut montrer que, pour un code raccourci par un motif S symétrique, les distances partielles, notées $D[S]_i$, correspondent aux poids des lignes de la matrice de transformation. Pour cela rappelons que les bits raccourcis sont des bits codés dont les valeurs sont contraintes à $x_i = 0$. Dans le cas d'un motif de raccourcissement symétrique, cela revient à imposer à $u_i = 0$ pour tous les bits impliqués dans le calcul de x_i, d'après l'équation (2.24):

$$x_i = \sum_{G_N^{(i)}[N-1-j]=1} u_j, \tag{B.11}$$

Autrement dit, la matrice de transformation d'un code raccourci par un motif symétrique est obtenue en supprimant les colonnes d'indices i tels que $S(i) = 1$, puis toutes les lignes ayant au moins un 1 sur l'une de ces colonnes (il s'avère qu'il s'agit des lignes d'indices i également, mais cela n'est pas utile pour cette étape de la démonstration). Dès lors, il est clair que les lignes conservées dans la matrice de

transformation $G[S]_N$ gardent le même poids que dans la matrice G_N. Or la matrice G_N vérifie la propriété suivante [58]:

$$D_i = d_H(G_N^{(i)}, \langle G_N^{(i+1)}, \ldots, G_N^{(N-1)} \rangle) = |G_N^{(i)}| = 2^{|B_n^{(i)}|}$$

Il s'ensuit finalement que:

$$D[S]_i = D_i = |G_N^{(i)}| = |G[S]_N^{(i)}|$$

□

C Décodage a inversion

C.1 Bornes idéales sur canal BEC

Il s'agit de dériver une expression analytique pour les bornes d'ordre ω, notés iWER-ω ,pour le canal à effacement. Cette expression exploite la classification des motifs de poinçonnages introduite dans le Chapitre 2.

Tout d'abord, en supposant le mot de code est $x_0^{N-1} = \mathbf{0}_N$, l'ordre d'une réalisation de bruit est donné par:

$$\omega_Y = |\mathcal{E}_Y| = |\{i \in \mathcal{I}, u_i^{(ga)} \neq 0\}|, \tag{C.1}$$

où $u_i^{(ga)}$ est la décision prise d'après le signe du LLR genie-aided $L_i^{(ga)}$, *i.e.* lorsque les positions précédentes sont connues. Sur un canal BEC et pour un décodeur genie-aided, on a $L_i^{(ga)} \geq 0$, [1] de sorte que l'ordre d'une réalisation de bruit pour un code $\mathcal{C}(N,K,\mathcal{I})$ est lié au nombre de fois où $L_i^{(ga)} = 0$ avec $i \in \mathcal{I}$.

Toute réalisation de bruit $\mathbf{Y}$ sur un canal BEC peut être assimilé à un motif de poinçonnage, déterminé par les positions des effacements introduits par le canal. Pour toute réalisation de bruit $\mathbf{Y}$, on note par $E[\mathbf{Y}]$, le motif des bits d'information effacés, défini par $E[\mathbf{Y}] \stackrel{\text{def}}{=} \{i = 0,\dots,N-1 \mid \mathrm{L}_i^{(ga)} = 0\}$. Par analogie avec le poinçonnage, on a $|E[\mathbf{Y}]| = |\mathbf{Y}|$. De plus, il apparait clair que si deux réalisations de bruits $\mathbf{Y}$ et $\mathbf{Y}'$ sont telles que $E[\mathbf{Y}] = E[\mathbf{Y}']$, alors celles-ci sont équivalentes du point de vue du décodeur SC pour le canal BEC. Notons que la notion de réalisations de bruit équivalentes sur le canal BEC est plus forte que la notion d'équivalence de deux motifs de poinçonnage, explicitée dans le Chapitre 2. En effet, deux réalisations de bruit d'un canal le BEC ayant le même motif d'effacement sont bien équivalentes, tandis qu'il a été montré dans le Chapitre 2 que deux motifs de poinçonnage peuvent avoir le même motif d'effacement sans être équivalents. Dès lors, considérer uniquement les motifs symétriques est représentatif de tous les résultats de décodage distincts.

Considérons un code polaire $\mathcal{C}(N,K,\mathcal{I})$ sur un canal BEC de probabilité d'effacement ϵ, et définissons $P_e(\lambda,\epsilon,\mathcal{C}) \stackrel{\text{def}}{=} \Pr(|E[\mathbf{Y}] \cap \mathcal{I}| = \lambda)$ la probabilité d'avoir exactement λ effacements sur les bits d'information. Puisque le décodeur prend une décision aléatoire lorsque le LLR est nul, il s'ensuit que:

$$\text{iWER-}\omega = \sum_{\lambda=\omega+1}^{K} (1 - 2^{\omega-\lambda}) \cdot P_e(\lambda,\epsilon,\mathcal{C}) \tag{C.2}$$

La valeur de $P_e(\lambda,\epsilon,\mathcal{C})$ peut être déterminée en comptant le nombre de réalisations de bruit telles que $|E[\mathbf{Y}] \cap \mathcal{I}| = \lambda$, où chaque réalisation est pondérée par sa probabilité donnée par $\epsilon^{|\mathbf{Y}|}(1-\epsilon)^{N-|\mathbf{Y}|}$. En exploitant l'analogie avec le poinçonnage, il s'ensuit que $P_e(\lambda,\epsilon,\mathcal{C})$ peut être évaluée en énumérant

[1] Le cas $L_i^{(ga)} = -\infty$ ne peut pas survenir pour un décodeur genie-aided avec le mot de code $x_0^{N-1} = \mathbf{0}_N$, du fait que si deux LLRs λ_a et λ_b sont positifs, alors $\mathrm{BP}(\lambda_a,\lambda_b) \geq 0$ (où $\mathrm{BP}(.)$ correspond à la fonction de mise à jour du nœud xor, donnée par l'équation (1.32)) et $\lambda_a + \lambda_b \geq 0$. Par récurrence cette propriété, vérifiée sur l'entrée du décodeur, se propage à l'ensemble des LLRs du graphe.

TABLE C.1: *Énumération des motifs symétriques (lorsque* $|E[\mathbf{Y_S}] \cap \mathcal{I}| > 0$*)*

motif	$\lvert\mathbf{Y_S}\rvert$	$\lvert E[\mathbf{Y_S}] \cap \mathcal{I}\rvert$	Multiplicité
$[1,0,1,0,1,0,\mathbf{1},0]$	4	1	2
$[1,1,0,0,1,\mathbf{1},0,0]$	4	1	4
$[1,1,1,0,1,0,\mathbf{1},0]$	5	1	8
$[1,1,1,0,1,\mathbf{1},0,0]$	5	1	16
$[1,1,1,\mathbf{1},0,0,0,0]$	4	1	16
$[1,1,1,\mathbf{1},1,0,0,0]$	5	1	32
$[1,1,1,0,1,\mathbf{1},\mathbf{1},0]$	6	2	4
$[1,1,1,\mathbf{1},1,0,\mathbf{1},0]$	6	2	8
$[1,1,1,\mathbf{1},1,\mathbf{1},0,0]$	6	2	16
$[1,1,1,\mathbf{1},1,\mathbf{1},\mathbf{1},0]$	7	3	8
$[1,1,1,\mathbf{1},1,\mathbf{1},\mathbf{1},\mathbf{1}]$	8	4	1

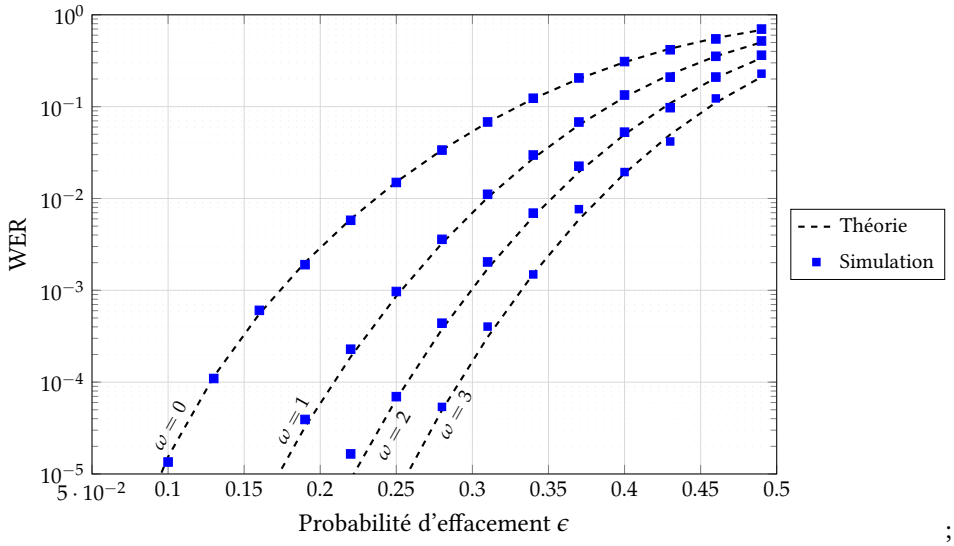
uniquement les motifs symétriques, ainsi que leurs multiplicités, notées $M(\mathbf{Y_s})$, correspondant au nombre de motifs ayant pour motif d'effacement le motif $E[\mathbf{Y}]$, . Le résultat donne la formule suivante:

$$\begin{aligned} P_e(\lambda,\epsilon,\mathcal{C}) &= \Pr(|E[\mathbf{Y}] \cap \mathcal{I}| = \lambda) \\ &= \sum_{\mathbf{Y}_s \in \mathbb{S}} \Pr(\mathbf{Y}_s) \cdot \Pr(|E[\mathbf{Y}_S] \cap \mathcal{I}| = \lambda \mid \mathbf{Y}_s) \\ &= (1-\epsilon)^N \sum_{\substack{\mathbf{Y}_s \in \mathbb{S} \\ |E[\mathbf{Y}_s] \cap \mathcal{I}| = \lambda}} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathbf{Y}_s|} \cdot M(\mathbf{Y_s}) \end{aligned}$$

où $\mathbb{S}$ est l'ensemble des motifs symétriques.

Nous fournissons un exemple pour le code $(N,K) = (8,4)$ avec $\mathcal{I} = \{3,5,6,7\}$. La liste des motifs symétriques tels que $|E[\mathbf{Y_S}] \cap \mathcal{I}| > 0$ est donnée dans le tableau C.1. On en déduit que (on notera $x = \frac{\epsilon}{1-\epsilon}$):

$$\begin{aligned} P_e(1,\epsilon,\mathcal{C}) &= (1-\epsilon)^8 \cdot [22x^4 + 56x^5] \\ P_e(2,\epsilon,\mathcal{C}) &= (1-\epsilon)^8 \cdot [28x^6] \\ P_e(3,\epsilon,\mathcal{C}) &= (1-\epsilon)^8 \cdot [8x^7] \\ P_e(4,\epsilon,\mathcal{C}) &= (1-\epsilon)^8 \cdot [x^8] \end{aligned}$$



;

FIGURE C.1: *Bornes d'ordre ω pour un code $N = 64$, $K = 32$*

D'où:

$$\text{iWER-0} = (1-\epsilon)^8 \cdot [11x^4 + 28x^5 + 21x^6 + 7x^7 + \frac{15}{16}x^8]$$
$$\text{iWER-1} = (1-\epsilon)^8 \cdot [14x^6 + 6x^7 + \frac{7}{8}x^8]$$
$$\text{iWER-2} = (1-\epsilon)^8 \cdot [4x^7 + \frac{3}{4}x^8]$$
$$\text{iWER-3} = (1-\epsilon)^8 \cdot [\frac{1}{2}x^8]$$

Finalement, la figure C.1 illustre la correspondance parfaite entre les résultats issus de la formule (C.2) et les simulations pour un code $(N,K) = (64,32)$.

D Codes Polaires Non-binaires

D.1 Démonstration de la Proposition 23

On se propose de démontrer la proposition 23, à savoir que lorsque le noyau est positionné sur le premier étage du graphe (et ses entrées sont donc obtenues directement du canal), et en supposant un canal AWGN, alors la probabilité $\overline{\mathrm{SEP}_R^{\pi}}$ peut s'exprimer en fonction de la distance euclidienne sur la constellation.

Démonstration (Proposition 23) :

Reprenons de l'équation (5.11):

$$\overline{\mathrm{SEP}_R^{\pi}} = \frac{1}{|\mathcal{A}|^2} \sum_{u_0 \in \mathcal{A}} \sum_{(x_1, x_1') \in \mathcal{A}_*^2} P\left(u_0 + \pi^{-1}(x_1'), x_1'; u_0 + \pi^{-1}(x_1), x_1\right) \tag{D.1}$$

où $\mathcal{A}_*^2 = \left\{(x_1, x_1') \in \mathcal{A}^2 \mid x_1 \neq x_1'\right\}$ et $P\left(u_0 + \pi^{-1}(x_1'), x_1'; u_0 + \pi^{-1}(x_1), x_1\right)$ est la probabilité que (x_0', x_0) soit plsu probable que (x_0, x_1) sachant que $(X_0, X_1) = (x_0, x_1)$.

Soit $x' = (x_1', x_2') \in \mathcal{A}^2$ et $x = (x_0, x_1) \in \mathcal{A}^2$. Considérons que x est associé à un symbole de la constellation $\mathcal{X}^{2p/m}$ par la fonction d'attribution φ, et transmis ensuite sur un canal AWGN de variance σ^2. Le symbole reçu $y \in \mathbb{K}^{2p/m}$ est donné par:

$$y = \varphi(x) + \zeta,$$

où $\zeta = (z_1, \ldots, z_{2p/m}) \in \mathbb{K}^{2p/m}$, avec $z_i \sim \mathcal{N}(0, \sigma^2)$. A l'issue de la démodulation, on obtient:

$$\begin{aligned} \lambda(a) &:= \lambda_0(a_0)\lambda_1(a_1) \\ &= \Pr(x = a \mid y) = \mu \cdot \exp\left(-\frac{\|y - \varphi(a)\|^2}{\kappa\sigma^2}\right), \ \forall a = (a_0, a_1) \in \mathcal{A}^2, \end{aligned}$$

où $\kappa = 2$ si $\mathbb{K} = \mathbb{R}$ (constellation réelle) ou $\kappa = 1$ si $\mathbb{K} = \mathbb{C}$ (constellation complexe), et μ est un facteur de normalisation, tel que $\sum_{a \in \mathcal{A}} \lambda(a) = 1$.

Ainsi, $P\left(x_0', x_1'; x_0, x_1\right)$ peut se ré-écrire:

$$\begin{aligned} P\left(x_0', x_1'; x_0, x_1\right) &= \Pr\left(\|y - \varphi(x')\|^2 \leq \|y - \varphi(x)\|^2\right) \\ &= \Pr\left(\|\theta + \zeta\|^2 \leq \|\zeta\|^2\right) \end{aligned}$$

où $\theta = \varphi(x) - \varphi(x')$. Soit $\theta = (t_1, \ldots, t_{2p/m}) \in \mathbb{K}^{2p/m}$. Alors:

$$\begin{aligned} \|\theta\|^2 &= \textstyle\sum_{j=1}^{2p/m} |t_j|^2 \\ \|\zeta\|^2 &= \textstyle\sum_{j=1}^{2p/m} |z_j|^2 \\ \|\theta + \zeta\|^2 &= \textstyle\sum_{j=1}^{2p/m} |t_j + z_j|^2 = \sum_{j=1}^{2p/m} \left(|t_j|^2 + |z_j|^2 + 2\langle t_j, z_j \rangle_{\mathbb{R}}\right) \end{aligned}$$

où $\langle t,z\rangle_{\mathbb{R}}$ correspond au produit scalaire de $t,z\in\mathbb{K}$, lorsque $\mathbb{K}$ est considéré comme un espace vectoriel réel. Ainsi $\langle t,z\rangle_{\mathbb{R}}=tz$, si $\mathbb{K}=\mathbb{R}$, et $\langle t,z\rangle_{\mathbb{R}}=\text{real}(t)\text{real}(z)+\text{imag}(t)\text{imag}(z)$, si $\mathbb{K}=\mathbb{C}$, et l'on obtient alors :

$$\|\theta+\zeta\|^2=\|\theta\|^2+\|\zeta\|^2+2\langle\theta,\zeta\rangle_{\mathbb{R}},$$

avec $\langle\theta,\zeta\rangle_{\mathbb{R}}:=\sum_{j=1}^{2p/m}\langle t_j,z_j\rangle_{\mathbb{R}}$. Il s'ensuit que:

$$P\left(x_0',x_1';x_0,x_1\right)=\Pr\left(\|\theta\|^2+2\langle\theta,\zeta\rangle_{\mathbb{R}}\leq 0\right)=\Pr\left(\langle\theta,\zeta\rangle_{\mathbb{R}}\leq-\frac{\|\theta\|^2}{2}\right)$$

Puisque $\langle\theta,\zeta\rangle_{\mathbb{R}}$ suit une loi normale de moyenne nulle et de variance $\sigma_\theta^2=\frac{2}{\kappa}\|\theta\|^2\sigma^2$, avec κ définit précédemment ($\kappa=2$ si $\mathbb{K}=\mathbb{R}$, où $\kappa=1$ si $\mathbb{K}=\mathbb{C}$), l'expression précédente devient:

$$\begin{aligned}P\left(x_0',x_1';x_0,x_1\right)&=\Pr\left(\frac{\langle\theta,\zeta\rangle_{\mathbb{R}}}{\sigma_\theta}\leq-\frac{\sqrt{\kappa}\|\theta\|}{2\sqrt{2}\sigma}\right)\\&=Q\left(\frac{\sqrt{\kappa}\|\theta\|}{2\sqrt{2}\sigma}\right)\end{aligned}$$

avec $D_{\mathcal{E}}(x_0',x_1';x_0,x_1)=\|\theta\|$ la distance euclidienne entre (x_0',x_1') et (x_0,x_1). En remplaçant cette expression dans l'expression de $\overline{\text{SEP}_R^{\pi}}$, on obtient finalement :

$$\overline{\text{SEP}_R^{\pi}}=\frac{1}{|\mathcal{A}^2|}\sum_{u_0\in\mathcal{A}}\sum_{(u_1,u_1')\in\mathcal{A}_*^2}Q\left(\frac{\sqrt{\kappa}}{2\sqrt{2}\sigma}D_{\mathcal{E}}(x_0',x_1';x_0,x_1)\right)\tag{D.2}$$

D.2 Démonstration de la Proposition 24

Démonstration (Proposition 24) : On considère, pour chaque $w=1,\ldots,p-1$, les deux ensembles suivants:

$$\begin{aligned}S_w^-&=\{(a,b)\in\mathcal{A}_*^2\mid|a+b|\leq w\}\\S_{p-w}^+&=\{(a,b)\in\mathcal{A}_*^2\mid|a+b|>p-w\}\end{aligned}$$

dont le nombre d'éléments se calcule par:

$$\begin{aligned}|S_w^-|&=q\sum_{k=1}^{w}\binom{p}{k}\\|S_{p-w}^+|&=q\sum_{k=p-w+1}^{p}\binom{p}{k}=q\sum_{k=0}^{w-1}\binom{p}{k}\end{aligned}$$

On vérifie facilement que $|S_{p-w}^+|<|S_w^-|,\forall w=1,\ldots,p-1$.

Puisque $|S_{p-w}^+|<|S_w^-|$, il existe au moins un couple $(a,b)\in S_w^-$, tel que $(\pi(a),\pi(b))\notin S_{p-w}^+$. Des lors, $|a+b|\leq w$ et $|\pi(a)+\pi(b)|\leq p-w$, et par conséquent:

$$|a+b|^2+|\pi(a)+\pi(b)|^2\leq w^2+(p-w)^2=2w^2-2pw+p^2$$

Il s'ensuit que:

$$D_{min}^{(\pi)}\leq\min_{w=1,\ldots,p-1}2\sqrt{2w^2-2pw+p^2}$$

On vérifie facilement que pour p pair le minimum ci-dessus est égal a $\sqrt{2}p$ (et est obtenu pour $w=\frac{p}{2}$), alors que pour p impair le minimum ci-dessus est égal a $\sqrt{2}\sqrt{p^2+1}$ (et est obtenu pour $w=\frac{p-1}{2}$ ou $w=\frac{p+1}{2}$)

Bibliographie

[1] 3rd Generation Partnership Project (3GPP). *Multiplexing and channel coding*. 3GPP TS 38.212 V.15.3.0, Sept. 2018.

[2] Ahmed Abdmouleh et al. "A new approach to optimise Non-Binary LDPC codes for Coded Modulations". In: *9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*. IEEE. 2016, pp. 295–299.

[3] Orion Afisiadis, Alexios Balatsoukas-Stimming, and Andreas Burg. "A low-complexity improved successive cancellation decoder for polar codes". In: *48th Asilomar Conference on Signals, Systems and Computers*. 2014, pp. 2116–2120.

[4] Amin Alamdar-Yazdi and Frank R Kschischang. "A simplified successive-cancellation decoder for polar codes". In: *IEEE communications letters* 15.12 (2011), pp. 1378–1380.

[5] Erdal Arikan. "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels". In: *IEEE Transactions on Information Theory* 55.7 (2009), pp. 3051–3073.

[6] E Arikan et al. "Performance of short polar codes under ML decoding". In: *ICT-Mobile Summit Conf. Proc.* 2009.

[7] Valentin Bakoev. "One more way for counting monotone boolean functions". In: *Thirteenth International Workshop on Algebraic and Combinatorial Coding Theory*. 2012, p. 4752.

[8] Alexios Balatsoukas-Stimming, Mani Bastani Parizi, and Andreas Burg. "LLR-based successive cancellation list decoding of polar codes". In: *IEEE transactions on signal processing* 63.19 (2015), pp. 5165–5179.

[9] Magali Bardet et al. "Algebraic properties of polar codes from a new polynomial formalism". In: *IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 230–234.

[10] Mani Bastani Parizi. "Polar Codes: Finite Length Implementation, Error Correlations and Multilevel Modulation". MA thesis. Swiss Federal Institute of Technology, 2012.

[11] Meryem Benammar et al. "Multi-kernel polar codes: Proof of polarization and error exponents". In: *Information Theory Workshop (ITW)*. IEEE. 2017, pp. 101–105.

[12] Guillaume Berhault. "Exploration architecturale pour le décodage de codes polaires". PhD thesis. Université de Bordeaux, 2015.

[13] Elwyn R Berlekamp and NJ A_ Sloane. "Restrictions on weight distribution of Reed-Muller codes". In: *Information and Control* 14.5 (1969), pp. 442–456.

[14] Valerio Bioglio, Carlo Condo, and Ingmar Land. "Design of Polar Codes in 5G New Radio". In: *arXiv preprint arXiv:1804.04389* (2018).

[15] Valerio Bioglio, Frederic Gabry, and Ingmar Land. “Low-Complexity Puncturing and Shortening of Polar Codes”. In: *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2017, pp. 1–6.

[16] Valerio Bioglio et al. “Minimum-Distance Based Construction of Multi-Kernel Polar Codes”. In: *arXiv preprint arXiv:1701.07616* (2017).

[17] Sarit Buzaglo et al. “On Efficient Decoding of Polar Codes with Large Kernels”. In: *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2017, pp. 1–6.

[18] Giuseppe Caire, Giorgio Taricco, and Ezio Biglieri. “Bit-interleaved coded modulation”. In: *IEEE Transactions on Information Theory* 44.3 (1998), pp. 927–946.

[19] Ludovic Chandesris, Valentin Savin, and David Declercq. “An Improved SCFlip for Polar Codes”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2016, pp. 1–6.

[20] Ludovic Chandesris, Valentin Savin, and David Declercq. “Dynamic-SCFlip decoding of polar codes”. In: *IEEE Transactions on Communications* (2018).

[21] Kai Chen, Kai Niu, and Jia-Ru Lin. “An efficient design of bit-interleaved polar coded modulation”. In: *IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. 2013, pp. 693–697.

[22] Kai Chen, Kai Niu, and Jia-Ru Lin. “Practical polar code construction over parallel channels”. In: *IET Communications* 7.7 (2013), pp. 620–627.

[23] Kai Chen, Kai Niu, and Jiaru Lin. “Improved successive cancellation decoding of polar codes”. In: *IEEE Transactions on Communications* 61.8 (2013), pp. 3100–3107.

[24] Peiyao Chen, Baoming Bai, and Xiao Ma. “A New Construction of Nonbinary Polar Codes with Two-stage Polarization”. In: *arXiv preprint arXiv:1801.08059* (2018).

[25] Sae-Young Chung et al. “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit”. In: *IEEE Communications letters* 5.2 (2001), pp. 58–60.

[26] Gabriele Coppolino et al. “A Multi-Kernel Multi-Code Polar Decoder Architecture”. In: *arXiv preprint arXiv:1802.00580* (2018).

[27] Jincheng Dai et al. “Does gaussian approximation work well for the long-length polar code construction?” In: *IEEE Access* 5 (2017), pp. 7950–7963.

[28] David Declercq and Marc Fossorier. “Decoding algorithms for nonbinary LDPC codes over GF (q)”. In: *IEEE Transactions on Communications* 55.4 (2007), pp. 633–643.

[29] Peter Elias. “Coding for two noisy channels”. In: *IRE International Convention Record*. 1955, pp. 37–46.

[30] Ahmed Elkelesh et al. “Belief propagation decoding of polar codes on permuted factor graphs”. In: *Wireless Communications and Networking Conference (WCNC)*. IEEE. 2018, pp. 1–6.

[31] Furkan Ercan, Carlo Condo, and Warren J Gross. “Improved Bit-Flipping Algorithm for Successive Cancellation Decoding of Polar Codes”. In: *arXiv preprint arXiv:1808.03616* (2018).

[32] Furkan Ercan et al. “Partitioned Successive-Cancellation Flip Decoding of Polar Codes”. In: *International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–6.

[33] Ubaid U Fayyaz and John R Barry. “Low-complexity soft-output decoding of polar codes”. In: *IEEE Journal on Selected Areas in Communications* 32.5 (2014), pp. 958–966.

[34] Arman Fazeli and Alexander Vardy. “On the scaling exponent of binary polarization kernels”. In: *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2014, pp. 797–804.

[35] G David Forney Jr. “Codes on graphs: normal realizations”. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 520–548.

[36] RG Gallagher. *Low-density parity-check codes, no. 21 in Research Monograph Series*. 1963.

[37] Pascal Giard and Andreas Burg. “Fast-SSC-flip decoding of polar codes”. In: *Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE. 2018, pp. 73–77.

[38] Pascal Giard, Claude Thibeault, and Warren J Gross. *High-speed decoders for polar codes*. Springer, 2017.

[39] Talha Cihad Gulcu, Min Ye, and Alexander Barg. “Construction of polar codes for arbitrary discrete memoryless channels”. In: *IEEE Transactions on Information Theory* 64.1 (2018), pp. 309–321.

[40] Jing Guo and Albert Guillen i Fabregas. “Efficient sphere decoding of polar codes”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2015, pp. 236–240.

[41] Seyyed Ali Hashemi, Carlo Condo, and Warren J Gross. “Simplified successive-cancellation list decoding of polar codes”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 815–819.

[42] Seyed Hamed Hassani, Kasra Alishahi, and Rüdiger L Urbanke. “Finite-length scaling for polar codes”. In: *IEEE Transactions on Information Theory* 60.10 (2014), pp. 5875–5898.

[43] Gaoning He et al. “β-expansion: A Theoretical Framework for Fast and Recursive Construction of Polar Codes”. In: *arXiv preprint arXiv:1704.05709* (2017).

[44] Zhiliang Huang et al. “On the Successive Cancellation Decoding of Polar Codes with Arbitrary Linear Binary Kernels”. In: *arXiv preprint arXiv:1701.03264* 2.1 (2017), p. 0.

[45] J Huber and Udo Wachsmann. “Capacities of equivalent channels in multilevel coding schemes”. In: *Electronics Letters* 30.7 (1994), pp. 557–558.

[46] Nadine Hussami, Satish Babu Korada, and Rüdiger Urbanke. “Performance of polar codes for channel and source coding”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2009, pp. 1488–1492.

[47] Hideki Imai and Shuji Hirakawa. “A new multilevel coding method using error-correcting codes”. In: *IEEE Transactions on Information Theory* 23.3 (1977), pp. 371–377.

[48] Corina I Ionita et al. “On the design of binary polar codes for high-order modulation”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2014, pp. 3507–3512.

[49] Onurcan İşcan, Ronald Böhnke, and Wen Xu. “Shaped Polar Codes for Higher Order Modulation”. In: *IEEE Communications Letters* 22.2 (2018), pp. 252–255.

[50] Onurcan İşcan and Wen Xu. “Polar Codes with Integrated Probabilistic Shaping for 5G New Radio”. In: *arXiv preprint arXiv:1808.09360* (2018).

[51] Rolf Johannesson and Kamil Sh Zigangirov. *Fundamentals of convolutional coding*. Vol. 15. John Wiley & Sons, 2015.

[52] Sinan Kahraman and Mehmet Ertugrul Celebi. “Code based efficient maximum-likelihood decoding of short polar codes”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2012, pp. 1967–1971.

[53] Daniel Kern, Sebastian Vorkoper, and Volker Kuhn. “A new code construction for polar codes using min-sum density”. In: *8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*. 2014, pp. 228–232.

[54] Jaeyoel Kim, Jong-Hwan Kim, and Sang-Hyo Kim. “An efficient search on puncturing patterns for short polar codes”. In: *International Conference on Information and Comm. Tech. Convergence (ICTC)*. 2015, pp. 182–184.

[55] Satish Babu Korada, Eren Sasoglu, and Rüdiger Urbanke. “Polar codes: Characterization of exponent, bounds, and constructions”. In: *IEEE Transactions on Information Theory* 56.12 (2010), pp. 6253–6264.

[56] Jeffrey C Lagarias. “The Takagi function and its properties”. In: *arXiv preprint arXiv:1112.4205* (2011).

[57] Stephane Le Goff, Alain Glavieux, and Claude Berrou. “Turbo-codes and high spectral efficiency modulation”. In: *IEEE International Conference on Communications*. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE). 1994, pp. 645–645.

[58] Myung-Kyu Lee and Kyeongcheol Yang. “The exponent of a polarizing matrix constructed from the Kronecker product”. In: *Designs, codes and cryptography* 70.3 (2014), pp. 313–322.

[59] Camille Leroux et al. “Hardware architectures for successive cancellation decoding of polar codes”. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 1665–1668.

[60] Hsien-Ping Lin, Shu Lin, and Khaled Abdel-Ghaffar. “Binary nonlinear kernels of maximum exponents of polar codes of dimensions up to sixteen”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2015, pp. 221–225.

[61] David JC MacKay and Radford M Neal. “Near Shannon limit performance of low density parity check codes”. In: *Electronics letters* 32.18 (1996), pp. 1645–1646.

[62] Vera Miloslavskaya. “Shortened polar codes”. In: *IEEE Transactions on Information Theory* 61.9 (2015), pp. 4852–4865.

[63] Vera Miloslavskaya and Peter Trifonov. “Sequential decoding of polar codes”. In: *IEEE Communications Letters* 18.7 (2014), pp. 1127–1130.

[64] Marco Mondelli, S Hamed Hassani, and Rüdiger Urbanke. “Construction of Polar Codes with Sublinear Complexity”. In: *arXiv preprint arXiv:1612.05295* (2016).

[65] Marco Mondelli, S Hamed Hassani, and Rudiger L Urbanke. “From polar to Reed-Muller codes: a technique to improve the finite-length performance”. In: *IEEE Transactions on Communications* 62.9 (2014), pp. 3084–3091.

[66] Marco Mondelli, S Hamed Hassani, and Rüdiger L Urbanke. “Scaling exponent of list decoders with applications to polar codes”. In: *IEEE Transactions on Information Theory* 61.9 (2015), pp. 4838–4851.

[67] Marco Mondelli, S Hamed Hassani, and Rüdiger L Urbanke. “Unified scaling of polar codes: Error exponent, scaling exponent, moderate deviations, and error floors”. In: *IEEE Transactions on Information Theory* 62.12 (2016), pp. 6698–6712.

[68] Ryuhei Mori and Toshiyuki Tanaka. “Channel polarization on q-ary discrete memoryless channels by arbitrary kernels”. In: *International Symposium on Information Theory Proceedings (ISIT)*. IEEE. 2010, pp. 894–898.

[69] Ryuhei Mori and Toshiyuki Tanaka. “Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes”. In: *Information Theory Workshop (ITW)*. IEEE. 2010, pp. 1–5.

[70] Ryuhei Mori and Toshiyuki Tanaka. “Performance and construction of polar codes on symmetric binary-input memoryless channels”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2009, pp. 1496–1500.

[71] David E Muller. “Application of Boolean algebra to switching circuit design and to error detection”. In: *Transactions of the IRE Professional Group on Electronic Computers* 3 (1954), pp. 6–12.

[72] K Niu and K Chen. “Stack decoding of polar codes”. In: *Electronics letters* 48.12 (2012), pp. 695–697.

[73] Kai Niu, Kai Chen, and Jia-Ru Lin. “Beyond turbo codes: rate-compatible punctured polar codes”. In: *International Conference on Communications (ICC)*. 2013, pp. 3423–3427.

[74] Kai Niu et al. “Rate-Compatible Punctured Polar Codes: Optimal Construction Based on Polar Spectra”. In: *arXiv preprint arXiv:1612.01352* (2016).

[75] Charly Poulliat, Marc Fossorier, and David Declercq. “Design of regular (2, d/sub c/)-LDPC codes over GF (q) using their binary images”. In: *IEEE Transactions on Communications* 56.10 (2008).

[76] Noam Presman, Ofer Shapira, and Simon Litsyn. “Mixed-Kernels Constructions of Polar Codes”. In: *IEEE Journal on Selected Areas in Communications* 34.2 (2015), pp. 239–253.

[77] Tobias Prinz et al. “Polar coded probabilistic amplitude shaping for short packets”. In: *18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. 2017, pp. 1–5.

[78] Minghai Qin et al. “Polar Code Constructions Based on LLR Evolution”. In: *IEEE Communications Letters* (2017).

[79] Irving S Reed and Gustave Solomon. “Polynomial codes over certain finite fields”. In: *Journal of the society for industrial and applied mathematics* 8.2 (1960), pp. 300–304.

[80] Thomas J Richardson and Rüdiger L Urbanke. “The capacity of low-density parity-check codes under message-passing decoding”. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 599–618.

[81] Steven Roman. *Coding and information theory*. Vol. 134. Springer Science & Business Media, 1992.

[82] Hamid Saber and Ian Marsland. “An incremental redundancy hybrid ARQ scheme via puncturing and extending of polar codes”. In: *IEEE Transactions on Communications* 63.11 (2015), pp. 3964–3973.

[83] Gabi Sarkis et al. “Fast list decoders for polar codes”. In: *IEEE Journal on Selected Areas in Communications* 34.2 (2016), pp. 318–328.

[84] Gabi Sarkis et al. “Fast polar decoders: Algorithm and implementation”. In: *IEEE Journal on Selected Areas in Communications* 32.5 (2014), pp. 946–957.

[85] Eren Şaşoğlu, Emre Telatar, and Erdal Arikan. “Polarization for arbitrary discrete memoryless channels”. In: *Information Theory Workshop (ITW)*. IEEE. 2009, pp. 144–148.

[86] Philipp Schläfer et al. “A new architecture for high throughput, low latency NB-LDPC check node processing”. In: *26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE. 2015, pp. 1392–1397.

[87] Christian Schürch. “A partial order for the synthesized channels of a polar code”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 220–224.

[88] Mathis Seidl et al. “Aspects of polar-coded modulation”. In: *9th International ITG Conference on Systems, Communication and Coding (SCC), Proceedings o*. VDE. 2013, pp. 1–6.

[89] Mathis Seidl et al. “Polar-coded modulation”. In: *IEEE Transactions on Communications* 61.10 (2013), pp. 4108–4119.

[90] Claude E Shannon and Warren Weaver. *The Mathematical Theory of Information (Urbana, IL)*. 1949.

[91] Dong-Min Shin, Seung-Chan Lim, and Kyeongcheol Yang. “Design of length-compatible polar codes based on the reduction of polarizing matrices”. In: *IEEE Transactions on Communications* 61.7 (2013), pp. 2593–2599.

[92] Dong-Min Shin, Seung-Chan Lim, and Kyeongcheol Yang. “Mapping selection and code construction for 2ˆ m-ary polar-coded modulation”. In: *IEEE Communications Letters* 16.6 (2012), pp. 905–908.

[93] Norbert Stolte. “Recursive codes with the Plotkin-Construction and their Decoding”. PhD thesis. Ph. D. dissertation, University of Technology Darmstadt, Germany.

[94] Ido Tal and Alexander Vardy. “How to construct polar codes”. In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6562–6582.

[95] Ido Tal and Alexander Vardy. “List decoding of polar codes”. In: *IEEE International Symposium on Information Theory (ISIT)*. 2011, pp. 1–5.

[96] Ido Tal and Alexander Vardy. “List decoding of polar codes”. In: *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2213–2226.

[97] Saurabha R Tavildar. “Bit-permuted coded modulation for polar codes”. In: *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2017, pp. 1–6.

[98] Peter Trifonov. “Efficient design and decoding of polar codes”. In: *IEEE Transactions on Communications* 60.11 (2012), pp. 3221–3227.

[99] Peter Trifonov. “Reduced complexity decoding of polar codes with Reed-Solomon kernel”. In: ().

[100] Peter Trifonov, Vera Miloslavskaya, and Ruslan Morozov. “Fast Sequential Decoding of Polar Codes”. In: *arXiv preprint arXiv:1703.06592* (2017).

[101] Grigorii Trofimiuk and Peter Trifonov. “Block sequential decoding of polar codes”. In: *International Symposium on Wireless Communication Systems (ISWCS)*. 2015, pp. 326–330.

[102] Gottfried Ungerboeck. “Channel coding with multilevel/phase signals”. In: *IEEE Transactions on Information Theory* 28.1 (1982), pp. 55–67.

[103] Udo Wachsmann, Robert FH Fischer, and Johannes B Huber. “Multilevel codes: Theoretical concepts and practical design rules”. In: *IEEE Transactions on Information Theory* 45.5 (1999), pp. 1361–1391.

[104] Runxin Wang and Rongke Liu. “A novel puncturing scheme for polar codes”. In: *IEEE Communications Letters* 18.12 (2014), pp. 2081–2084.

[105] Tao Wang, Daiming Qu, and Tao Jiang. “Parity-check-concatenated polar codes”. In: *IEEE Communications Letters* 20.12 (2016), pp. 2342–2345.

[106] Niclas Wiberg, Hans-Andrea Loeliger, and Ralf Kotter. “Codes and iterative decoding on general graphs”. In: *European Transactions on telecommunications* 6.5 (1995), pp. 513–525.

[107] McReynolds Wozencraft and B. Reiffen. *Sequential Decoding*. M.I.T Press. 1961.

[108] Peihong Yuan and Fabian Steiner. “Efficient Decoding Algorithms for Polar Codes based on 2\times2 *Non — BinaryKernels*”. In: *arXiv preprint arXiv:1807.03767* (2018).

[109] Liang Zhang et al. “On the puncturing patterns for punctured polar codes”. In: *International Symposium on Information Theory*. 2014, pp. 121–125.

[110] Yingxian Zhang et al. “A Practical Construction Method for Polar Codes”. In: *IEEE Communication Letter* 18.11 (2014).

[111] Zhaoyang Zhang et al. “Progressive Bit-Flipping Decoding of Polar Codes Over Layered Critical Sets”. In: *Global Communications Conference (Globecom)*. IEEE. 2017, pp. 1–6.