THÈSE de DOCTORAT

---

présentée par

Antoine VALEMBOIS

pour obtenir le grade de

Docteur de l'Université de Limoges

Discipline : Mathématiques et Applications
Spécialité : Mathématiques et Informatique

Décodage, Détection et Reconnaissance des Codes Linéaires Binaires

---

Soutenue le 20 Octobre 2000,

devant le jury composé de :

Mme	Pascale	Charpin	Codirecteur
M.	Nicolas	Sendrier	Codirecteur
M.	Thierry	Berger	Codirecteur
M.	John T.	Coffey	Rapporteur
M.	Sami	Harrari	Rapporteur
M.	François	Laubie	Examinateur

[...] vous, ô mathématiques concises, par l'enchaînement rigoureux de vos propositions tenaces et la constance de vos lois de fer, vous faites luire, aux yeux éblouis, un reflet puissant de cette vérité suprême dont on remarque l'empreinte dans l'ordre de l'univers.
Comte de Lautréamont (Maldoror, Chant deuxième)

Il n'est jamais problème qui n'ait un cadeau pour toi entre ses mains. Tu cherches des problèmes parce que tu as besoin de leurs cadeaux.
Richard Bach (Illusions)

Remerciements

Je dois le fait d'avoir pu m'exprimer à travers cette thèse, et d'avoir passé ces trois années dans le milieu de la recherche, à Thierry Berger, mon professeur de codage en D.E.A., qui en a trouvé l'opportunité et m'a jugé digne d'en profiter.

Je tiens à remercier sincèrement, et de manière générique, toutes les personnes qui m'ont aidé intentionnellement. Certaines, je le soupçonne, l'ont fait à mon insu et en restant dans l'ombre. Je ne les nommerai pas d'une part parce qu'il ne s'agit que de soupçons, d'autre part pour respecter leur discrétion bien que je n'en connaisse pas la raison. Je souhaite simplement qu'elles se reconnaissent et reçoivent ces remerciements.

Je n'aurais pas pu mener ce travail à bien sans l'atmosphère propice à la recherche et à la créativité qui règne au *projet Codes* de l'INRIA. Ses membres permanents, Nicolas Sendrier, Daniel Augot et Anne Canteaut, en ont constamment assuré le bon fonctionnement infrastructurel, et par là même éloigné de moi les préoccupations non directement liées à mes recherches.

Ils ont toujours été de bon conseil, et leurs conversations m'ont plongé au coeur d'une problématique passionnante. En particulier, si ce travail présente quelque rigueur, c'est à Nicolas Sendrier que je le dois, qui a contribué de manière essentielle à corriger dans ce sens mes considérations par trop intuitives, imprécises et pour ainsi dire incommunicables.

Mais c'est vers Pascale Charpin, directrice du *projet*, que se porte spontanément le plus gros de ma reconnaissance. Elle supervise, par petites touches, discrètement mais savamment appliquées et sans la moindre ingérence, les potentialités de chacun.

En particulier, sa fine psychologie, sa prudence éprouvée et un grand discernement ont préservé ma vie estudiantine des difficultés que j'ai pu connaître dans ma vie privée, ce qu'un chaperonnage indiscret n'eût pas permis. Elle m'a fait confiance, a été humainement irréprochable, cela m'a été d'un précieux secours, et je l'en remercie du fond du coeur.

Bien entendu, je remercie aussi le *projet Codes* pour m'avoir envoyé, en tant qu'auditeur ou conférencier, à de prestigieux colloques, et pour m'avoir fait rencontrer des personnes pour lesquelles j'ai beaucoup d'admiration; en un mot, pour m'avoir fait membre d'une communauté scientifique que je brûlais de rejoindre.

François Sirven, ingénieur au laboratoire *Traitement de Guerre Electronique* de *Thomson CSF communication* responsable de la bonne adéquation de mes travaux aux problèmes qui se posent réellement, m'a beaucoup aidé par sa grande compétence, son travail généreux et passionné, et son intelligence à laquelle j'ai été fort sensible. Nos contacts auront sans-doute été trop rares par ma faute, mais suffisants pour que ces travaux soit empreints de son pragmatisme et aient pris le bon cap dès le départ. Il a fait preuve d'une grande patience et d'autant de souplesse et de gentillesse, que je n'ai pas ménagées. Je tiens à lui dire ma gratitude et à ce qu'il sache qu'il est pour beaucoup dans la réussite de cette thèse.

Par ailleurs, j'ai découvert dans la recherche un métier et je l'espère une

vocation, mais encore fallait-il y croire, et ne pas être seul à y croire. De ce point de vue, les contacts que j'ai eus avec Ilya Dumer et Marc Fossorier à *ISIT 2000* ont été pour moi fort gratifiants et m'ont donné un regain de confiance et d'énergie en fin de thèse, sans lequel ce manuscrit aurait sans-doute moins fière allure.

Enfin, j'ai reçu un précieux soutien d'amis et connaissances dont je ne peux dresser ici une liste exhaustive; je citerai tout de même celles et ceux qui m'ont hébergé et dont j'ai pu partagé l'hygiène de vie: Marie-Pierre, Loïc et le trio Anne / Xavier / Philippe; ainsi que ceux qui m'ont enrichi d'un peu de leur érudition exaltée: Fabrice Pautot, Pascal Orosco et Fabien N'Guyen Huu.

A tous, un grand merci.

Résumé

Trois problèmes liés à la transmission d'un signal numérique et aux codes correcteurs d'erreur sont étudiés, seul le cas du train binaire avec ou sans information souple et des codes linéaires binaires est traité.

Le premier et le plus classique de ces trois problèmes, le décodage, a été abondamment étudié par le passé, et sous des angles fort différents. Dans cette étude, l'objectif prioritaire est la meilleure utilisation possible du canal. La complexité du décodage vient ensuite.

Pour cette raison, nous étudierons les codes linéaires binaires généraux (aléatoires) de grande longueur, qui offrent les meilleures propriétés spectrales; et leur décodage à quasi-maximum de vraisemblance qui permet d'atteindre les taux d'erreur les plus bas tout en coûtant moins cher que le décodage complet.

L'ambition de mettre au point *l'algorithme le plus efficace* pour venir à bout du problème (NP-complet) posé par ce décodage a continuellement guidé les choix bibliographiques et l'angle d'étude. Finalement, une solution est proposée qui semble effectivement la meilleure à ce jour.

Les deux autres problèmes: la détection et la reconnaissance de code sont en revanche originaux en ce sens que très peu de littérature leur est consacrée. Ils consistent, en observant un train binaire issu d'une transmission sur un canal bruité, à déterminer si un code linéaire binaire a été utilisé pour porter le signal (détection), et si oui, lequel (reconnaissance).

On montrera que le problème est NP-complet, et l'on considèrera ses différents aspects. Une étude est ensuite menée sur la bonne manière de résoudre une catégorie plus vaste de problèmes comprenant ces deux là: les problèmes de détection et de reconnaissance.

Un certain pragmatisme accompagne la mise au point de solutions, en même temps qu'un souci d'optimalité et de généralité. Mais ces solutions sont par nature perfectibles et ne seront suggérées qu'après avoir donné les outils qui permettront à tout un chacun de mettre au point sa propre solution.

Avertissement

D'aucuns, qui me feront toutefois l'honneur de me lire, trouveront prétentieux de ma part d'avoir inséré au début de chaque chapitre d'illustres citations littéraires. Non pas qu'ils pensent sérieusement que «je mets côte à côte mes écrits et ceux de Molière ou Shakespeare parce que j'estime qu'ils sont d'essence comparable» (la vrai raison est bien-sûr que «tous ces auteurs peuvent être pour moi des sources d'inspiration que je souhaiterais partager»), mais le simple fait d'avoir envisagé cette idée monstrueuse, même s'ils l'ont immédiatement rejetée, perturbe encore leur jugement.

D'autres trouveront cela simplement indécent, j'accepte plus volontiers cette critique.

Mais écrire, même une thèse, dans le but d'être lu, n'est-il pas toujours prétentieux et indécent ? En écrivant, on se révèle au lecteur, ce qui est indécent; et l'on entend par là l'enrichir ou le changer, ce qui est prétentieux.

Il me semble quant à moi que je suis moins à même que les auteurs que je cite, d'enrichir ou de changer le lecteur, qui je l'espère, daignera s'attarder sur leurs vénérables pensées, pour y découvrir, créer du sens. Ce n'est pas pour une autre raison que leurs esprits sont invoqués, et cette substitution me paraît être le contraire d'indécence et prétention.

Quoi qu'il en soit, la liberté de juger ne saurait être enfreinte par cet avertissement; je m'en remets au lecteur avec abandon.

Table des matières

Remerciements **3**

Résumé **5**

Avertissement **6**

Table des figures **11**

Introduction générale **13**

I Décodage **15**

Introduction **17**

1 Généralités **19**

Introduction 19

1.1 Problématique 20

1.1.1 Les origines 20

1.1.2 Capacités du canal gaussien 23

1.1.3 Codes linéaires binaires, définitions 27

1.1.4 Taux d'erreur 30

1.2 Décodage (quasi) optimal 31

1.2.1 De l'intérêt des codes aléatoires 31

1.2.2 Décodage complet 32

1.2.3 Décodage quasi-complet 33

1.2.4 Critère d'arrêt 34

1.3 Ordre exponentiel 35

1.3.1 Définitions et propriétés 35

1.3.2 Probabilité de succès, quasi-complétude, nombre d'itérations 36

Conclusion 37

2 Décodage dur **39**
Introduction . . . 39
2.1 Décodage par recherche du motif d'erreur . . . 40
2.1.1 Complexité . . . 41
2.1.2 Restriction à un ensemble d'information . . . 41
2.1.3 Séparation en deux des mots de l'espace ambiant . . . 42
2.1.4 Utilisation de codes poinçonnés . . . 43
2.2 Décodage par ensemble d'information . . . 46
2.2.1 Principe et justification . . . 46
2.2.2 Description d'une itération, format des données . . . 47
2.2.3 Pivot vicinal . . . 48
2.2.4 Complexité . . . 50
2.3 Décodage mixte . . . 52
2.3.1 Recherche de motifs d'erreur: approche directe . . . 52
2.3.2 Utilisation de codes poinçonnés . . . 54
2.3.3 Utilisation de surcodes . . . 56
2.4 Pivots vicinaux exclusivement . . . 59
2.4.1 modélisation par un processus markovien . . . 62
2.4.2 Résultats explicites dans un cas particulier . . . 66
2.4.3 Cas où le coset leader n'est pas unique . . . 69
Conclusion . . . 70

3 Décodage souple **71**
Introduction . . . 71
3.1 contexte et notations . . . 72
3.1.1 Canal, modulation et alphabet . . . 73
3.1.2 Variables aléatoires . . . 74
3.1.3 Calcul des probabilités, maximum de vraisemblance et distance généralisée . . . 75
3.2 Outils spécifiques au décodage souple . . . 77
3.2.1 Critère d'arrêt . . . 77
3.2.2 Tri des positions par ordre de fiabilité . . . 78
3.2.3 Enumération ordonnée des motifs d'erreur . . . 80
3.3 Décodage par ensemble d'information souple . . . 81
3.3.1 Problématique . . . 81
3.3.2 Décodage par recouvrement d'ellipsoïdes . . . 82
3.3.3 Décodage par résonnance stochastique . . . 84
3.4 Performances . . . 85
Conclusion . . . 86

Conclusion **91**

II Détection et reconnaissance **93**

Introduction **95**

1 Histoires de rang **97**
Introduction . . . 97
1.1 Critère du rang . . . 98
1.1.1 Probabilité de fausse alarme . . . 98
1.1.2 Probabilité de détection . . . 99
1.2 Oracle du rang . . . 100
1.2.1 Remarques sur la longueur et la synchronisation . . . 100
1.2.2 Remarques sur la dimension . . . 101
1.3 Réduction de rang, NP-complétude du problème . . . 103
1.3.1 Notations et définition du problème . . . 104
1.3.2 Complexité . . . 104
Conclusion . . . 105

2 Détection et reconnaissance: généralités **107**
Introduction: inconnu et hasard . . . 107
2.1 Problèmes de détection . . . 108
2.1.1 Définition . . . 108
2.1.2 Critère de détection, probabilités de détection et de fausse alarme . . . 109
2.1.3 Détection optimale . . . 110
2.1.4 Détection optimale sous contrainte de temps . . . 113
2.2 Problèmes de reconnaissance . . . 117
2.2.1 Reconnaissance optimale . . . 117
2.2.2 Reconnaissance sous contrainte de temps . . . 117
Conclusion . . . 118

3 Détection optimale: Code de parité **119**
Introduction . . . 119
3.1 Cas dur . . . 120
3.1.1 Principe de l'algorithme de détection . . . 120
3.1.2 Variables aléatoires . . . 120
3.1.3 Calcul des probabilités . . . 121
3.2 Cas souple . . . 125
3.2.1 Variables aléatoires . . . 125
3.2.2 Calcul des probabilités . . . 126
3.3 Reconaissance de longueur et de synchronisation . . . 128
3.3.1 Cas dur . . . 129
3.3.2 Cas souple . . . 131
Conclusion . . . 132

4 Cas général: contrainte du temps **133**
Introduction . . . 133
4.1 Générateur de candidats . . . 134
4.1.1 Moments d'ordre supérieur . . . 134
4.1.2 Génération de moments de poids faible . . . 135
4.2 Critères . . . 137

4.2.1 Notations et remarques 138
4.2.2 Cas dur 140
4.2.3 Cas souple 145
4.2.4 Coûts de calcul des critères 148
4.3 Détection 149
4.3.1 Exemple d'algorithme 149
4.3.2 Cas général 160
4.3.3 Reconnaissance de longueur et de synchronisation 160
4.4 Reconnaissance de code 162
4.4.1 Estimation d'une hypothèse sous contrainte de temps . . 164
4.4.2 Incrémentation de la dimension courante 164
4.4.3 Décrémentation de la dimension courante 166
4.4.4 Exemple d'algorithme 169
Conclusion 170

Conclusion **173**

Conclusion générale **175**

Index **183**

Index **183**

Résumé **183**

Table des figures

1.1 Capacité des différents canaux 26
1.2 Exemple de courbe SNR 28

2.1 Coefficient de complexité des différents algorithmes 60

3.1 Modulation antipodal 73
3.2 Bruit additif 73
3.3 Résonnance stochastique: $n = 128, R = 1/2, E_b/N_0 = 2\,\mathrm{dB}$ 87
3.4 RS: $n = 128, R = 1/2, E_b/N_0 = 3\,\mathrm{dB}$ 87
3.5 RS: $n = 96, R = 1/2, E_b/N_0 = 4\,\mathrm{dB}$ 88
3.6 RS: $n = 180, R = 3/4, E_b/N_0 = 4\,\mathrm{dB}$ 88
3.7 Coefficient de complexité des algorithmes de décodage souple . . 89

2.1 Probabilité de détection en fonction de la probabilité de fausse alarme 110

4.1 Constellations des couples de poids $(\mathrm{wt}(h), \mathrm{wt}(hX^t))$ obtenu par l'algorithme du premier ordre pour un [64,34]-code 143

Introduction générale

Cette thèse fut le plaisir et la chance de trois années consacrées à me poser les questions essentielles et fondatrices de la *théorie de l'information*. Elle m'a offert le recul des quelques cinquante années de réponses apportées à ces questions, de synthèses et de créations qui ont enthousiasmé deux générations de chercheurs (et la troisième, qui écrit ses thèses, voit cela d'un bon oeil).

Enthousiasme parce qu'il s'agit là d'une discipline assez nouvelle, accessible et utile, ce qui n'est pas si fréquent; mais surtout parce que cette discipline est, à de beaux égards, intellectuellement exaltante: en ayant la pureté des mathématiques, mais également en donnant lieu, comme les sciences de la nature, à l'élaboration de modèles devant «coller» à une réalité empirique dont le détail infini nous échappe; ou encore, éventuellement, en effleurant métaphysiquement la question du *verbe* ou celle du *temps*.

Mais elle m'a surtout demandé de me construire une idée intuitive du train binaire, de ce que les lois des grands nombres ont à offrir à la dichotomie, à la dualité, à la dialectique, à l'opposition, à la «non unité». Le train binaire a ses lois, dictées par le hasard, sa mécanique statistique. Ses éléments, finis, binaires, obéissent en groupe aux exigences de l'infini, et leur demander d'obéir individuellement à d'autres exigences coûte, quoi qu'il en soit, de l'énergie, ne serait-ce qu'un quantum.

La théorie de l'information fait l'étude de ces lois, de cette énergie, dans le but de l'économiser et d'accroître les possibilités du train binaire, avec pour matière première la combinatoire, et pour ouvrière l'algorithmique.

Il aurait pu s'agir d'étudier le détail d'abstractions mathématiques fantastiques, aux propriétés algébriques puissantes, manipulables avec virtuosité par l'algorithmique, qui ont représenté un centre d'intérêt de premier ordre pour la recherche durant ce demi-siècle.

Il s'est agi au contraire de bien mesurer les limites de l'algorithmique, de se contenter de juste assez de structure algébrique pour pouvoir utiliser une représentation concise, d'ignorer les singularités aux bonnes propriétés pour se consacrer à une vaste classe d'objets auxquels un maximum d'autres puissent se ramener. Sur ce terrain, au sein de cette généralité, l'algorithmique est d'une nature exponentielle, et l'on s'accorde avec bon sens (même si aucune preuve ne semble vouloir se dévoiler) à dire que c'est incontournable.

Cette thèse fut avant tout, et sera peut-être pour certains lecteurs malgré les limites de ce manuscrit, en quelque sorte une descente en profondeur dans le

monde des codes correcteurs d'erreurs, et plus particulièrement de l'association train binaire codé linéairement / bruit gaussien; une appropriation par l'intuition de ce qui différencie ce train binaire codé, malgré son couplage avec un phénomène stochastique et les distorsions qu'il pourra subir, d'un ensemble de tirages à pile ou face; et ce, dans le but de pouvoir extraire l'information que porte cette différence.

Elle fut donc également, et c'est essentiellement ce que l'on trouvera dans ce document, une étude des outils algorithmiques qui pourront nous y aider et une contribution à les perfectionner, et à repousser les limites que leur exponentialité nous impose (outrageusement), quitte à parfois combattre le mal par le mal, la stochasticité par la stochasticité.

Première partie

Décodage

Introduction

La problématique liée aux codes correcteurs a suscité et suscitera certainement encore longtemps une étonnante dépense d'énergie intellectuelle, d'argent et de trésors de subtilité scientifique.

Il s'agit d'optimiser encore et toujours des concepts liés à la qualité d'une transmission d'information et de déterminer les limites que l'on ne pourra théoriquement jamais dépasser. Citons, entre autres concepts, le taux d'erreur, le délai, le coût, le débit, la quantité de ressource physique requise (énergie ou largeur de bande de fréquence par exemple)...

Notre époque offre plus que jamais une foule de motivations pour améliorer cette qualité: l'internet, la téléphonie, les médias, les satellites, les sondes interplanétaires... La transmission d'information a toujours eu une importance fondamentale dans la société, mais les mutations que celle-ci a subies exigent des performances infiniment supérieures à celles qui étaient atteintes grâce aux tambours ou aux nuages de fumée.

Le codage n'est bien sûr que l'une des nombreuses techniques en jeu dans cette problématique et la plupart des gens ignorent tout simplement son existence, de même la théorie de l'information est moins enseignée que le traitement du signal ou l'électromagnétisme. Pourtant la quête d'optimalité tendra à fondre ces différentes techniques en un procédé global. La succession verticale des différentes disciplines s'aplatira irrémédiablement et une expertise globale sera de plus en plus requise pour permettre les progrès.

Mais au delà de l'aspect technique et d'intérêts pécuniers faramineux, le défi scientifique est fondamental, abstrait et concret semblant se rapprocher l'un de l'autre, en un point de rencontre métaphysiquement fascinant. Ma conviction est que le point de convergence de la nature et de la connaissance se situe dans le chaos, la stochasticité, l'indéterminisme; elles se rejoignent à l'extrémité imaginaire de leurs ramifications infinies.

De ce point de vue, la théorie de l'information et plus particulièrement le codage de canal offre un cadre exemplaire pour s'approprier intuitivement une image de cette inaccessible perfection. C'est un peu dans le but d'offrir une illustration, même chétive, de ces propos par trop lyriques, que cette partie sur le décodage se terminera sur le concept de résonnance stochastique, aboutissement de mes travaux de thèse (puisqu'il faut bien s'arrêter), et je l'espère, ouverture sur de prochaines investigations scientifiques.

Chapitre 1

Généralités

Les idées s'améliorent. Le sens des mots y participe. Le plagiat est nécessaire. Le progrès l'implique. Il serre de près la phrase d'un auteur, se sert de ses expressions, efface une idée fausse, la remplace par l'idée juste.

(citation plagiée par Guy Debord, la société du spectacle, thèse 207) *Isidore Ducasse (Poésies II)*

C'est une grande jouissance que de se transporter dans l'esprit des temps passés, de voir comme un sage a pensé avant nous, et comment, partis de loin, nous l'avons si victorieusement dépassé.

Johann Wolfgang von Goethe (Faust)

[...] nous ne connaissons jamais que les passions des autres, et ce que nous arrivons à savoir des nôtres, ce n'est que d'eux que nous avons pu l'apprendre.

Marcel Proust (Du côté de chez Swann)

Introduction

Nous allons donner dans ce chapitre une courte synthèse de la problématique de la transmission d'information numérique par un canal. Il n'est pas nécessaire d'assimiler cette synthèse pour lire la suite du manuscrit. Il n'y sera fait référence que sporadiquement au travers de remarques qualitatives.

Une justification de l'utilité des codes linéaires binaires pour la transmission d'information en découle naturellement. Les codes linéaires binaires sont des objets mathématiques simples dont on donnera la définition en section 1.1.3, et qu'il faut impérativement bien connaître pour lire la suite du manuscrit.

Il est donc conseillé au lecteur profane en matière de codes de commencer

par ouvrir l'un des nombreux ouvrages sur le sujet (par exemple [Bar98], [CC81] ou les très classiques [Gal68] et [MS77]); il pourra se contenter des passages traitant des codes linéaires binaires pour l'appréhension desquels un premier cycle universitaire en mathématiques suffit amplement.

On dépeindra quelques généralités sur leur décodage dans un contexte de transmission. Celles-ci seront valables aussi bien pour le décodage dur que pour le décodage souple. Lorsque l'on parlera de « distance », de « coset-leader », de « poids » ou de « rayon de recouvrement », il ne s'agira donc pas forcément d'une référence à la métrique de Hamming, habituellement considérée pour le décodage dur (voir le chapitre traitant du décodage souple pour la définition de métriques alternatives plus intéressantes).

1.1 Problématique

1.1.1 Les origines

Dans son remarquable article «A Mathematical Theory of Communication» de 1948 [Sha48], Claude E. Shannon fabrique les outils de ce qui sera la théorie de l'information. Il est à l'origine des notions d'entropie, d'information mutuelle et de capacité définies plus bas.

Son domaine d'étude est la transmission d'information numérique à travers un canal. Un canal relie une source à un destinataire, et chaque utilisation du canal consiste en la transmission d'un symbôle loisible pris dans un alphabet connu des deux protagonistes de la transmission. Le choix de l'alphabet et de la manière dont ses symbôles devront porter l'information (codage de canal) détermine aussi bien le débit maximal d'information que le niveau de confiance que le destinataire pourra accorder à l'information qu'il reçoit.

Le codage de canal est la composée du «codage binaire», tel qu'on le considère dans ce manuscrit, qui est de taux au plus 1, et du «codage binaire à signal» qui consiste à transformer un train de bits en un train de symbôles de l'alphabet d'entrée du canal et qui est de taux au moins 1. Le taux de codage de canal, produit de ces deux taux, est le rapport de nombre de bits du message à transmettre, au nombre d'utilisations du canal (il peut être supérieur à 1).

La capacité d'un canal peut être vue comme étant la limite, lorsque le nombre d'utilisations du canal tend vers l'infini, de la quantité maximale d'information qu'il puisse transporter avec un niveau de confiance maximal, divisée par le nombre d'utilisations. Encore faut-il définir une quantité «physique» extensive (additive) associée à la notion subjective d'information.

Le point de départ du travail de Shannon est l'énoncé suivant, souvent appelée *théorème du codage de canal bruité*:

Théorème 1 *Tout canal a une capacité C, et pour tout $R < C$, il existe des codages de canal de taux R qui, a l'aide d'un décodage à maximum de vraisemblance, permettent d'atteindre des taux d'erreur de transmission arbitrairement petit.*

Ce théorème est trivial en ce sens qu'il n'impose pas à la capacité d'être strictement positive. En fait la capacité du canal est définie comme étant le plus grand nombre C tel que le théorème ci-dessus soit vérifié, et Shannon, ayant eu l'intuition que ce nombre était palpable, cherchera à le calculer.

Définissons un canal sans mémoire par un triplet $(X, Y, P_{Y|X})$, où $X = (x_1 ... x_e)$ est appelé alphabet d'entrée, $Y = (y_1 ... y_s)$ alphabet de sortie, et $P_{Y|X} = (p_{j|i})_{j=1...s, i=1...e} \in [0,1]^{s \times e}$ matrice de probabilité de transition. Si l'on désigne par x et y respectivement les symboles d'entrée et de sortie du canal, considérés comme des variables aléatoires, la loi statistique du canal est donnée par $\Pr(y = y_j | x = x_i) = p_{j|i}$. $P_{Y|X}$ est donc une matrice stochastique (en colonne) et doit vérifier pour tout $i = 1...e$:

$$\sum_{j=1}^{s} p_{j|i} = 1$$

Si x suit une loi de probabilité $P_X = (\Pr(x = x_i))_{i=1...e} = (p_i)_{i=1...e}$, on définit l'entropie (binaire) de X comme étant la quantité:

$$H(X) \stackrel{def}{=} \Leftrightarrow \sum_{i=1}^{e} p_i \log_2 p_i$$

De P_X et $P_{Y|X}$, on peut déduire la distribution de probabilité jointe $P_{YX} = (p_{ji})_{j=1...s, i=1...e}$; $p_{ji} = \Pr(y = y_j, x = x_i) = p_i p_{j|i}$, et l'on définit l'entropie conditionnelle $H(Y|X)$ par:

$$H(Y|X) \stackrel{def}{=} \Leftrightarrow \sum_{i=1}^{e} \sum_{j=1}^{s} p_{ji} \log_2 \frac{p_{ji}}{p_i}$$

Ainsi que l'information mutuelle $I(X, Y)$ par

$$I(X,Y) \stackrel{def}{=} \sum_{i=1}^{e} \sum_{j=1}^{s} p_{ji} \log_2 \frac{p_{ji}}{p_i p_j}$$

Ces quelques outils ont une signification intuitive précise, à la manière des grandeurs «thermodynamiques», que ce manuscrit n'a pas pour vocation d'expliquer (on se reportera aux travaux de Shannon, ou au chapitre 2, *A measure of information*, de [Gal68]), et leur axiomatique donne lieu à toute une panoplie de formules dont voici juste deux exemples:

$$H(X,Y) \stackrel{def}{=} H(X) + H(Y|X) = H(Y) + H(X|Y) = H(Y,X)$$
$$I(X,Y) = H(X) \Leftrightarrow H(X|Y) = H(Y) \Leftrightarrow H(Y|X) = I(Y,X)$$

Se servant de ces outils, Shannon établit les performances maximales en terme de transmission d'information, qu'un canal bruité puisse atteindre. Il établit que:

Théorème 2 *la capacité C d'un canal (X, Y, P) est égale à son information mutuelle maximale:*

$$C = \max_X I(X, Y)$$

Le maximum étant pris sur toutes les distributions de probabilités possibles pour l'alphabet d'entrée X.

La démonstration qu'il en donne consiste en la coïncidence d'une minoration et d'une majoration de cette capacité. La minoration, basée sur ce que l'on appelle aujourd'hui un «code en bloc»[1] dont les mots sont tirés aléatoirement, et sur la loi faible des grands nombres, exhibe cet autre résultat (que j'adapte à la théorie moderne). qu'il n'a pas énoncé, faute sans doute, d'un vocabulaire adapté:

Théorème 3 *Pour tout canal de capacité C, il existe une fonction $\overline{E} :]0, C[\rightarrow \mathbf{R}^{+\star}$ telle que pour tout couple (K, n) d'entiers vérifiant $\frac{\log_2 K}{n} < C$, il existe des codes en bloc de longueur n sur l'alphabet d'entrée du canal, contenant K mots qui, a l'aide d'un décodage à maximum de vraisemblance, permettent d'atteindre des taux d'erreur de transmission inférieurs à $10^{-\overline{E}(\frac{\log_2 K}{n})n}$.*

Le plus faible taux d'erreur que l'on puisse atteindre avec un (K, n)-code est donc ici majoré. Pour obtenir ce résultat, il a suffi à Shannon de majorer le taux d'erreur binaire du meilleur code parmi le type de code particulier qu'il considérait (et en l'occurence les codes aléatoires se prêtent bien aux études statistiques).

Il s'avère qu'une minoration de ce même taux d'erreur (le plus faible que l'on puisse atteindre avec un (K, n)-code) est plus délicate à obtenir. Il faudra attendre 1959 [Sha59] pour que Shannon y parvienne dans le cas d'un canal gaussien et 1967 [SGB67] (avec l'aide de Gallager et Berlekamp) pour le cas d'un canal discret quelconque. J'introduis cependant ici ce théorème anachronique dont on aura besoin par la suite:

Théorème 4 *Pour tout canal de capacité C, il existe une fonction $\underline{E} :]0, C[\rightarrow R^{+\star}$ telle que pour tout $R \in]0, C[$, aucun code de longueur n sur l'alphabet d'entrée du canal et contenant plus de 2^{Rn} mots ne permette d'atteindre un taux d'erreur de transmission inférieur à $10^{-\underline{E}(R)n}$.*

Les théorèmes de 1948 sont remarquablement audacieux à cette époque où les codes correcteurs d'erreur n'existaient pas sous d'autres formes que celle d'une redondance d'alphabets tels que le morse (Shannon donne cependant un exemple plus efficace que le morse issu des premiers travaux de son collègue des *Bell Labs* R. W. Hamming, qui ne publiera quant à lui qu'en 1950 [Ham50]).

Mais le point de vue statistique et les preuves d'existence chères à Shannon ont peut-être soulevé les questions fondamentales de la théorie du codage

1. ensemble de $n-$uplet (dits les mots), pour n, la longueur de bloc, donné, d'éléments d'un alphabet donné de taille quelconque.

mais n'y répondent pas. Shannon répond aux questions « Quelles sont les performances que les meilleurs codes permettent d'atteindre ? », ou « Comment ces performances sont-elles reliées au choix des alphabets d'entrée et de sortie ? », mais les questions « Comment construire de bons codes ? » et « Comment décoder ces codes ? » sont laissées en suspens.

D'autres scientifiques se montrent toutefois inspirés par ces problèmes. Ainsi, P. Elias invente les codes convolutifs en 1955 [Eli55], et en 1956 D. Slepian (encore un ingénieur des *Bell Labs*) pose les fondations de la théorie des codes linéaires [Sle56]. Signalons aussi la trouvaille incroyablement précoce de M. J. E. Golay [Gol49], qui suite à la lecture de l'article de Shannon et bien avant les travaux de Slepian, invente sur le champ, dans une très courte et modeste correspondance, son fameux [23,12,7]-code linéaire binaire dont la perfection et la beauté combinatoire restent inégalées.

Depuis, les théoriciens de l'information et les ingénieurs des télécommunication ont essayé de réaliser les prédictions de Shannon, et s'en approchent progressivement. En 1995, on arrive avec les turbo-codes [BGT93] à obtenir un taux d'erreur binaire de 10^{-6} à un rapport signal à bruit supérieur d'à peine 0.7 dB à la limite de Shannon (cf. fin de cette section).

1.1.2 Capacités du canal gaussien

L'utilisation d'un canal physique en modulation de phase ou d'amplitude consiste à émettre à une phase loisible un signal d'amplitude loisible. On représente généralement ce signal dans le plan complexe, dit « espace des phases » (cf. section 3.1, p. 72), par un point de module l'amplitude et d'argument la phase. Le bruit, quant à lui, et par définition, aura pour effet de perturber ces deux grandeurs[1]. Il provient aussi bien du canal physique que de l'amplificateur et des autres composants électronique du récepteur (bruit thermique).

L'alphabet d'entrée du canal peut alors être vu comme l'ensemble des différents points de l'espace des phases que l'on s'autorise à émettre, et l'alphabet de sortie comme l'ensemble des différents points que l'on est capable de mesurer. L'énergie d'un signal est la moyenne du carré de son amplitude multiplié par sa durée. En pratique, ces alphabets sont finis, mais il est fréquent, pour ce qui est de la théorie, de considérer les limites, lorsqu'elles existent, de certaines grandeurs, telle par exemple que la capacité, lorsque la taille d'un alphabet tend vers l'infini.

Définition 1 *Le canal est dit sans mémoire (et le bruit, « blanc ») si les modifications subies par les signaux successifs sont statistiquement indépendantes. Il est dit additif si la différence entre le signal émis et le signal reçu est statistiquement indépendante du signal émis.*

Le modèle de canal le plus répandu est le canal BBAG[2], pour la simple raison qu'il est, dans de nombreux cas, possible de s'y ramener.

1. En réalité, un signal et sa perturbation ont une profondeur dans le temps, cette représentation suppose qu'ont eu lieu démodulation, filtrage adapté, et échantillonnage.

2. bruit blanc additif gaussien ; terme anglais : AWGN, additive white gaussian noise

Pour ce canal, la différence entre le signal émis et le signal reçu est une variable aléatoire complexe de loi normale $\mathcal{N}(0, N_0)$. Sa projection sur un axe réel est donc une variable aléatoire réelle de loi normale $\mathcal{N}(0, N_0/2)$. Puisque l'on s'intéresse souvent aux distorsions subies dans une direction donnée, on invoque souvent la « densité monolatérale » de bruit $N_0/2$. L'énergie du bruit, en revanche, est sa variance N_0 multiplié par la durée du signal.

Les parties réelles et imaginaires de ce bruit étant indépendantes, on se sert souvent séparément des parties réelles et imaginaires du signal comme de deux canaux identiques pour transmettre deux séquences de signaux réels indépendantes. L'énergie des signaux complexes est alors le double de celle des signaux réels, le taux de codage de canal est double également.

Afin de réduire les probabilités de transition, il s'agira, pour une taille d'alphabet d'entrée donnée, d'espacer au maximum les différents points de l'alphabet dans l'espace des phases, en se limitant à une puissance moyenne (énergie sur temps, moyenne du carré de l'amplitude) maximale donnée $\mathcal{E}^2$.

Un alphabet d'entrée binaire pourra être constitué de deux points de même amplitude $\mathcal{E}$ et en opposition de phase, on parle alors de modulation antipodale ou BPSK[1], le taux de «codage binaire à signal» est alors égal à 1, et les taux de codage et de codage de canal coïncident.

Si l'on utilise deux telles modulations séparément sur les parties réelles et imaginaire, on obtient alors une QPSK[2], ces deux modulations sont donc en quelque sorte équivalentes.

Le rapport brut signal à bruit est le ratio de l'énergie du signal par celle du bruit. Il est souvent plus intéressant d'exprimer par son logarithme une grandeur x sans dimension, tel le rapport de deux quantités de même dimension, de deux énergies par exemple. On lui donne, ce faisant, une dimension, c.a.d. un système d'unités pour la mesurer, la plus classique étant le décibel (dB): $10\log_{10}(x)$ est la mesure de x en dB.

Il est possible avec ce modèle de calculer la capacité du canal en fonction du rapport brut signal à bruit pour différents alphabets d'entrée et de sortie. Voici, sans démonstration, quelques capacités usuelles (elles découlent du théorème 2):

- Si ces deux alphabets sont binaires, le canal gaussien peut alors se modéliser comme un canal binaire symétrique de probabilité de transition τ égale à la probabilité pour que le bruit (réel) envoie un point d'abscisse réelle $\mathcal{E}$ vers un point d'abscisse négative, soit $\tau = \int_{-\infty}^{-\mathcal{E}} \mathcal{N}(0, N_0/2)(x)dx = Q\left(\sqrt{\frac{2\,\mathcal{E}^2}{N_0}}\right)$ [3]. Sa capacité vaut alors $1 + \tau\log_2\tau + (1 \Leftrightarrow \tau)\log_2(1 \Leftrightarrow \tau) = 1 \Leftrightarrow H_2(\tau)$.

- Si ces deux alphabets sont l'ensemble des nombres réels ou complexes (cas limite), et si les symboles d'entrée suivent une loi normale de variance

1. Binary Phase Shift Keying
2. Quaternary Phase Shift Keying
3. Q est la fonction $x \mapsto \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{t^2}{2}}dt$

$\mathcal{E}^2$ (Shannon a montré que, sous la contrainte de l'énergie, cette distribution était optimale; le canal ainsi défini est appelé canal gaussien non contraint), alors la capacité du canal vaut $\frac{1}{2}\log_2\left(1+2\frac{\mathcal{E}^2}{N_0}\right)$.

– Si l'alphabet d'entrée est binaire et l'alphabet de sortie réel, ce que nous considèrerons pour l'étude du décodage souple, alors la capacité est:

$$\sqrt{\frac{\mathcal{E}^2}{\pi N_0}}\int_{-\infty}^{\infty}\left(\Leftrightarrow\frac{\mathcal{E}^2}{N_0}\frac{(y\Leftrightarrow 1)^2}{\ln 2}e^{-\frac{\mathcal{E}^2}{N_0}(y-1)^2}\right.$$
$$\left.\Leftrightarrow\log_2\left(\frac{1}{2}\left(e^{-\frac{\mathcal{E}^2}{N_0}(y-1)^2}+e^{-\frac{\mathcal{E}^2}{N_0}(y+1)^2}\right)\right)\frac{1}{2}\left(e^{-\frac{\mathcal{E}^2}{N_0}(y-1)^2}+e^{-\frac{\mathcal{E}^2}{N_0}(y+1)^2}\right)\right)dy$$

Ces trois capacités sont tracées sur la figure 1.1. On voit qu'une sortie réelle permet de gagner environ 1.4 dB par rapport à une sortie binaire et qu'à de faibles rapports signal à bruit, une entrée binaire est à peu près équivalente à une entrée réelle (à de forts rapports signal à bruit, une entrée réelle permet de transmettre plus d'information dans chaque symbole, et d'obtenir une capacité supérieure à 1).

Le rapport utile signal à bruit E_b/N_0 est le rapport de l'énergie par bit d'information à l'énergie du bruit. L'énergie par bit d'information E_b est l'énergie par symbôle transmis divisée par le taux de codage de canal:

$$E_b=\mathcal{E}^2/R,\quad 10\log_{10}(E_b/N_0)=10\log_{10}(\mathcal{E}^2/N_0)\Leftrightarrow 10\log_{10}(R)$$

Considérer E_b plutôt que $\mathcal{E}^2$ apporte plus de consistance au modèle proposé, car ne nécessite pas, en fin de compte, de considérer les notions quelque peu informelles de taille d'alphabet et de «nombre d'utilisations du canal», entre lesquelles, selon le point de vue adopté, il peut y avoir un transfert de valeur, induisant une ambigüité pour la considération de l'énergie d'un symbôle $\mathcal{E}^2$. On peut en effet considérer n utilisations d'un canal d'alphabet de taille q et d'énergie moyenne $\mathcal{E}^2$, là où d'autres ne voient qu'une seule utilisation d'un canal d'alphabet de taille q^n et d'énergie moyenne $n\,\mathcal{E}^2$. Toute chose égale par ailleurs, les deux points de vue donnent en revanche la même énergie par bit d'information.

Ainsi, pour décrire l'efficacité d'un code (et d'un algorithme de décodage) donné on trace souvent le taux d'erreur binaire (son logarithme en base 10) en fonction du rapport utile signal à bruit en dB (cf. exemple figure 1.2), les courbes «SNR» [1] ainsi tracées sont toujours décroissantes.

Le fait de prendre en compte le rapport utile signal à bruit permet de comparer entre eux des codes de taux différents: à une même abscisse, la même dépense d'énergie est requise pour transmettre une quantité donnée d'information, les taux d'erreur (en ordonnée) peuvent être alors facilement comparés; à une même ordonnée, donc pour un même taux d'erreur, on voit en un coup d'oeil quel code sera le plus économique.

1. Signal to Noise Ratio

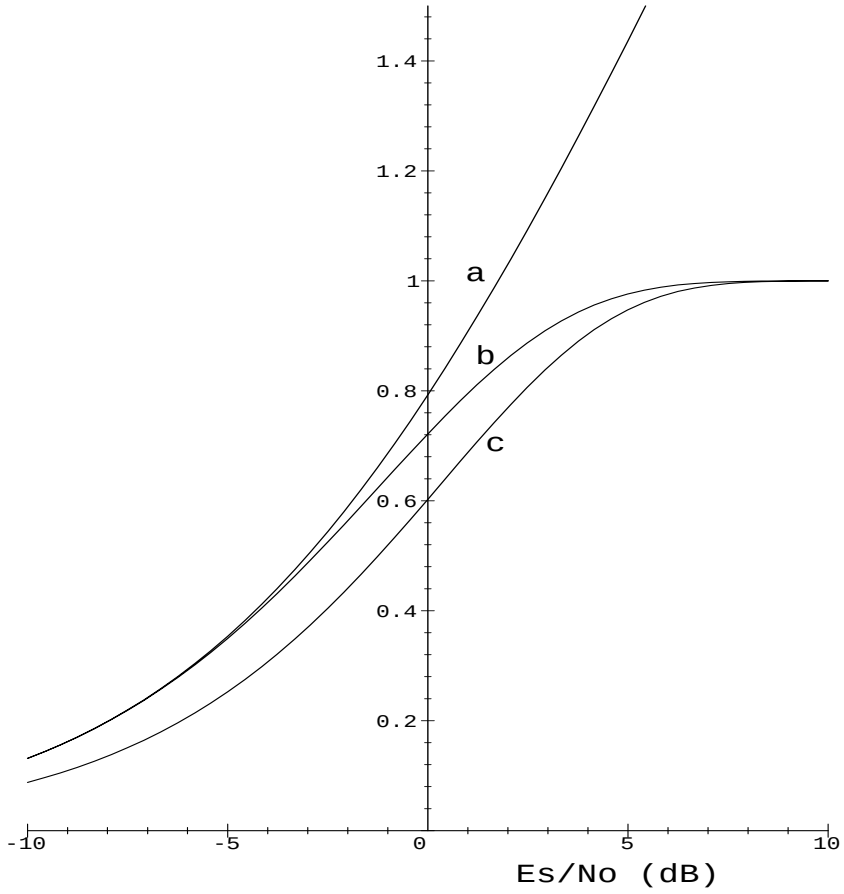


FIG. 1.1: *Capacité des différents canaux: (a) canal gaussien non contraint; (b) canal gaussien à entrée binaire et sortie réelle; (c) canal binaire symétrique*

Il s'agit de trouver un codage de canal (et un décodeur) de taux aussi grand que possible (pour avoir un débit d'information aussi élevé que possible), c'est-à-dire aussi proche que possible de la capacité, permettant d'atteindre des taux d'erreur aussi faibles que possible, à un rapport utile signal à bruit donné.

On voit sur la figure 1.2, tracée pour un canal BBAG et une modulation antipodale, que pour obtenir un taux d'erreur binaire de 10^{-3} à l'aide du [48,24,12]-code de Golay de taux $R = 1/2$ et d'un algorithme de décodage souple à maximum de vraisemblance, il suffit d'un rapport utile signal à bruit d'environ 3.6 dB (donc d'un rapport brut signal à bruit de 3.6 dB + $10\log_{10}(1/2) \approx 0.6$ dB), alors que sans codage (taux de codage égal à 1), le rapport signal à bruit doit être de 6.8 dB. On dit alors que le gain de codage est de 3.2 dB.

La capacité de ce canal (entrée binaire, sortie réelle) est tracée sur la figure 1.1. A un rapport brut signal à bruit de 0.6 dB, elle est de C=0.77=-1.15 dB, on dit que l'on transmet à $10\log[10](C/R) \approx$1.85 dB de la capacité (ce qui signifie que, théoriquement, avec un tel rapport brut signal à bruit, le gain de codage pourrait être d'autant supérieur).

Pour avoir une capacité égale au taux de codage $\frac{1}{2}$, il faudrait un rapport brut signal à bruit de -2.81 dB (le rapport utile signal à bruit correspondant, 0.19 dB, est représenté sur la figure 1.2 par une ligne verticale), c'est ce que l'on appelle la limite de Shannon. On voit donc aussi sur la figure que l'on est à 3.41 dB de la limite de Shannon (ce qui signifie que, théoriquement, avec un tel taux de codage, le gain de codage pourrait être d'autant supérieur).

Les courbes SNR's regroupe donc synthétiquement les informations quantitatives pertinentes sur l'efficacité d'un schéma de codage / décodage. Plus la courbe est basse et à gauche, meilleur est le schéma.

1.1.3 Codes linéaires binaires, définitions

Les codes que nous allons considérer tout au long de ce document sont donc les codes linéaires binaires, qui sont des cas particuliers des codes binaires en bloc. Un code en bloc est un ensemble de $n\Leftrightarrow$uplets de bits, où n est un entier appelé longueur du code. L'ensemble, que nous appellerons espace ambiant et que nous noterons $\mathbf{F}_2^n$, des $n\Leftrightarrow$uplets de bits, que nous appellerons les «mots», peut être muni d'une addition: l'addition modulo 2, bit à bit.

Celle-ci consititue évidemment une loi de groupe, tout mot est égal à son opposé pour cette loi; la différence et l'addition ne font qu'une. La multiplication par 0 ou 1 est trivial; l'ensemble des scalaires n'est constitué que d'un élément neutre et d'un élément absorbant pour cette multiplication, et l'espace ambiant possède donc une structure d'espace vectoriel.

Définition 2 *On appelle code linéaire binaire tout sous-espace vectoriel de l'espace ambiant $\mathbf{F}_2^n$, c.a.d. tout sous-ensemble de $\mathbf{F}_2^n$ stable par l'addition.*

En tant qu'espace vectoriel, un code possède donc une dimension, généralement notée k, des éléments, appelés mots de code, des bases, un code dual... Etant donnés k mots linéairement indépendants d'un code linéaire binaire de

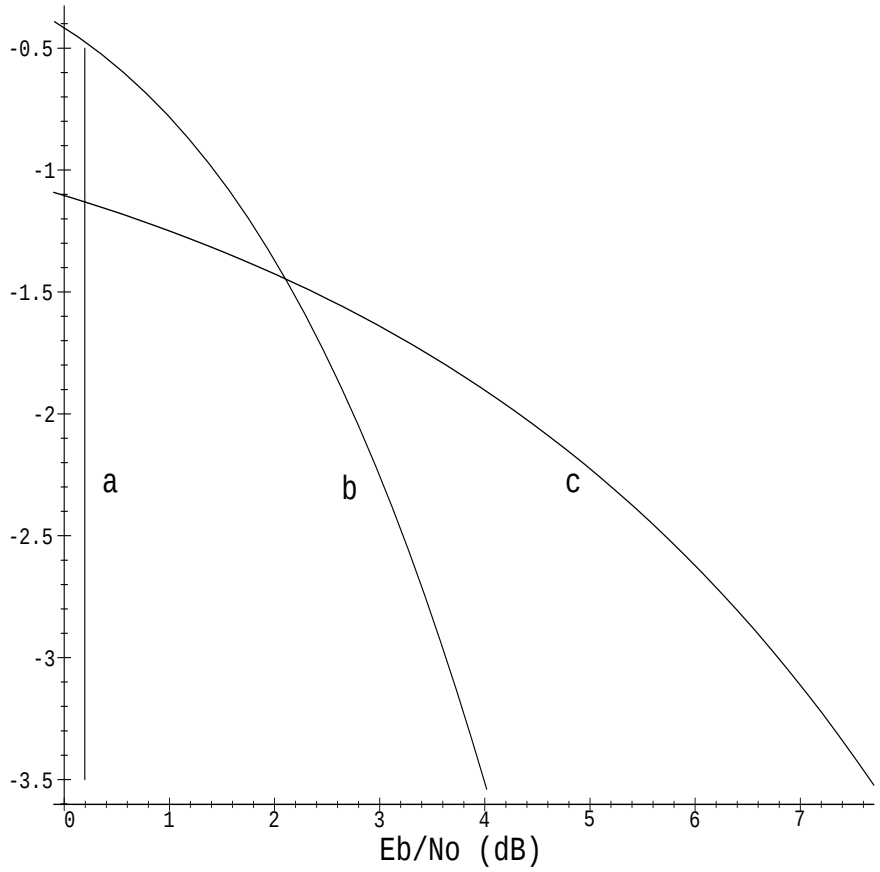


FIG. 1.2: *(a) Limite de Shannon pour $R = \frac{1}{2}$ sur le canal gaussien à entrée binaire; (b) courbe SNR du [48,24,12]-code de Golay avec décodage à maximum de vraisemblance; (c) taux d'erreur en l'absence de codage.*

dimension k (ils en consituent donc une base), la $k \times n$-matrice dont les lignes sont ces mots est appelée une matrice génératrice du code.

Une matrice génératrice du code dual d'un code est aussi appelée une matrice de parité du code. Ses lignes et toutes leurs combinaisons linéaires ont un produit scalaire nul avec tout mot de code.

En tant que sous-groupe, un code définit une relation d'equivalence sur l'espace ambiant.

Définition 3 *Un coset est un élément du quotient de l'espace ambiant par le code. Deux mots binaires appartiennent à un même coset si et seulement si leur différence appartient au code.*

Définition 4 *On appelle poids de Hamming d'un mot binaire, le nombre de ses bits égaux à 1. On appelle distance de Hamming entre deux mots le poids de Hamming de leur différence. On appelle distance minimale d'un code linéaire binaire, la plus petite distance de Hamming séparant deux mots de code, qui est aussi égale au plus petit poids de Hamming d'un mot de code.*

Définition 5 *Etant donnée une métrique sur l'espace ambiant (par exemple le poids de Hamming), on appelle coset leader (relativement à cette métrique) d'un mot ou de son coset, tout élément du coset qui minimise la métrique considérée.*

Pour la plupart des mots, on a unicité du coset leader et on utilisera souvent l'article défini pour parler d'un coset leader d'un mot.

Si un code linéaire binaire de dimension k est considéré, il contient 2^k mots. On appelle codage l'opération consistant à associer un mot de code à chaque k⇔uplet de bits. Pour ce faire, on emploie le plus souvent la multiplication à droite par une matrice génératrice du code. Si les k colonnes de gauche de la matrice génératrice considérée constutuent une matrice carrée diagonale, on dit que la matrice est systématique à gauche et que le codage est systématique, le k⇔uplet d'origine se retrouve alors dans le mot de code. Le taux de codage, rapport du nombre de bits d'information au nombre de symbôles binaires à transmettre, est égal à k/n.

Lorsqu'un mot de code traverse un canal, cette transmission peut inverser quelques bits du mot de code. La différence entre le mot émis et le mot reçu est appelée erreur de transmission. Puisque le mot émis est un mot de code, le mot reçu et l'erreur de transmission appartiennent à un même coset. Pour éliminer cette erreur, il faut trouver l'élément du coset du mot reçu qui lui correspond.

On espère généralement qu'il s'agit du coset leader, on définit d'ailleurs une métrique optimale sur l'espace ambiant de manière à ce que l'erreur la plus vraisemblable soit le coset leader. Pour un canal binaire symétrique, cette métrique optimale est le poids de Hamming. Nous appellerons décodage l'opération consistant à soustraire son coset leader au mot reçu[1].

1. usuellement, le décodage comprend aussi l'opération, à laquelle nous ne nous intéresserons pas, consistant à retrouver le $k-$uplet de bits correspondant au mot de code ainsi obtenu; pour un codage systématique, il suffit d'en prendre l'apocope de longueur k.

1.1.4 Taux d'erreur

Dans un contexte de transmission et avec la donnée d'un canal imparfait (bruité), le taux d'erreur résiduel après décodage (fréquence moyenne des erreurs considérées ou bien sur les bits (taux d'erreur binaire, t.e.b.) ou bien sur les mots de code (taux d'erreur de decodage, t.e.d.)), qu'il soit souple ou dur, n'est jamais nul. Cela est dû au fait qu'avec une probabilité P_{ML} non nulle, un motif d'erreur peut être «intrinsèquement indécodable» en ce sens qu'il ne sera pas égal à son propre coset leader.

Nous verrons par la suite, que pour dimensionner correctement les paramètres des algorithmes de décodage, aussi bien que pour majorer la dimension du code à détecter ou à reconnaître (deuxième partie) il est indispensable de pouvoir minorer *a priori* la probabilité d'erreur «intrinsèquement indécodable».

Le théorème 4 nous informe qu'il existe un nombre positif $\underline{E}$, fonction du taux de codage et de la capacité, tel que le taux d'erreur après décodage est supérieur à $10^{-\underline{E}n}$, il suffit alors de majorer $\underline{E}$. D'ailleurs, lorsqu'une minoration grossière du taux d'erreur est suffisante, il est fréquent que l'on se contente de considérer que, pour n suffisamment grand, $P_{ML} > n^{-n} = e^{-n \log n}$.

En revanche, dans un contexte de détection de code, lorsque le but de cette minoration est de majorer la dimension du code, mieux vaut être aussi fin que possible. Par ailleurs, sachant que le décodage qui sera appliqué par le destinataire ne sera probablement pas optimal, une approximation du taux d'erreur après un décodage optimal peut constituer une minoration du taux d'erreur effectif.

Pour cette raison, il est préférable de considérer les approximations proposée par Gregory Poltyrev [Pol94], bien qu'il s'agisse en réalité de majorations du plus faible taux d'erreur, car il semble qu'elle soit toujours très proche de ce taux d'erreur et en constitue une meilleure approximation que les minorations usuelles. La première de ces approximations, appelée «borne sphérique», s'applique au canal binaire symétrique et au décodage dur; la seconde, dite «borne tangentielle sphérique», s'applique au canal BBAG et au décodage souple (je l'énonce ci-dessous en ayant corrigé les coquilles présentes dans l'article de Poltyrev).

Ces deux bornes nécessitent la connaissance de la distribution des poids des mots de code. Lorsque celle-ci est inconnue, on pourra l'approximer par la distribution binômiale (on considère qu'il y a $\binom{n}{w}/2^{n-k}$ mots de poids w dans le code, même si cette quantité n'est pas un entier). J'ai calculé numériquement pour de nombreux codes les bornes obtenues dans ce cas et celles obtenues en considérant la distribution exacte, il s'est avéré que pour les codes de dimension supérieure à 50, la différence était toujours du même ordre que les erreurs d'arrondis, ou du moins était-elle invisible à l'oeil nu sur une courbe SNR.

Proposition 1 (Borne sphérique) *Soit C un $[n,k]$-code contenant C_w mots de poids w pour tout w; soit m le plus petit entier vérifiant:*

$$\sum_{w=1}^{2m} C_w \sum_{t=\lceil w/2 \rceil}^{m} \binom{w}{t} \binom{n \Leftrightarrow w}{m \Leftrightarrow t} \geq \binom{n}{m}$$

Alors, $P_{ML}^{(d)}$, le taux d'erreur de décodage après transmission par un canal binaire symétrique sans mémoire de probabilité de transition τ suivie d'un décodage complet, vérifie:

$$P_{ML}^{(d)} \leq \sum_{w=1}^{2(m-1)} C_w \sum_{t=\lceil w/2 \rceil}^{m-1} \binom{w}{t} \sum_{u=0}^{m-t-1} \binom{n \Leftrightarrow w}{u} \tau^{t+u} (1 \Leftrightarrow \tau)^{n-t-u} + \sum_{t=m}^{n} \binom{n}{t} \tau^t (1 \Leftrightarrow \tau)^{n-t}$$

Proposition 2 (Borne tangentielle sphérique) *Soit C un $[n,k]$-code contenant C_w mots de poids w pour tout w; Soit la fonction $\rho(x) = \frac{e^{-\frac{x^2}{N_0}\mathcal{E}^2}}{\sqrt{\pi N_0 / \mathcal{E}}}$; soit r_0 la racine de l'équation suivante en r:*

$$\sum_{w=1}^{\lceil \frac{rn}{r+n} \rceil - 1} C_w \int_0^{\arccos\left(\sqrt{\frac{nw}{r(n-w)}}\right)} (\sin\theta)^{n-3} d\theta = \frac{, \left(\frac{n-2}{2}\right)}{\sqrt{\pi},\ \left(\frac{n-1}{2}\right)}$$

(, $(z) = \int_0^\infty x^{z-1} e^{-x} dx$ étant la fonction d'Euler); et soit $R(z) = \frac{r_0}{n}(\sqrt{n} \Leftrightarrow z)^2$.

Alors, $P_{ML}^{(s)}$, le taux d'erreur de décodage après transmission par un canal BBAG de rapport brut signal à bruit $\frac{\mathcal{E}^2}{N_0}$ suivie d'un décodage à maximum de vraisemblance, vérifie:

$$P_{ML}^{(s)} \leq \int_{-\sqrt{n}}^{\sqrt{n}} \rho(z_1) \left(1 \Leftrightarrow \chi_{n-1}^2 \left(\frac{R(z_1)}{N_0/(2\,\mathcal{E}^2)} \right) + \sum_{w=1}^{\lceil \frac{r_0 n}{r_0+n} \rceil - 1} C_w \int_{\frac{\sqrt{n}-z_1}{\sqrt{\frac{n}{w}-1}}}^{\sqrt{R(z_1)}} \rho(z_2) \chi_{n-2}^2 \left(\frac{R(z_1) \Leftrightarrow z_2^2}{N_0/(2\,\mathcal{E}^2)} \right) dz_2 \right) dz_1$$

1.2 Décodage (quasi) optimal

1.2.1 De l'intérêt des codes aléatoires

Les codes les plus classiquement utilisés et étudiés sont ceux pour lesquels on parvient à mettre au point des algorithmes polynomiaux de décodage borné jusqu'à la «capacité de correction», qui, étant donné un mot de l'espace ambiant à distance inférieure à la «capacité de correction» d'un mot de code, retrouvent ce mot de code, à un coût polynomial en la longueur du code.

Ceci est rendu possible par la particularité qu'ont ces codes de permettre de décrire les cosets et leurs éléments dans une représentation à l'axiomatique plus puissante que celle des sou-espaces affines de $\mathbf{F}_2^n$.

Cette capacité de correction est généralement le rayon d'empilement du code, c.a.d. le plus grand rayon possible pour des boules disjointes centrées sur les éléments d'un même coset. S'il existe une solution au décodage borné jusqu'à ce nombre, elle est unique. Pour le décodage dur, il s'agit du plus grand entier strictement inférieur à la moitié de la distance minimale du code.

Ces algorithmes ne peuvent généralement qu'échouer lorsque le mot de l'espace ambiant qui leur est soumis est à distance supérieure à cette capacité de correction de tout mot de code. Pour que cet évènement soit rare, il est nécessaire de transmettre loin de la capacité du canal. A des taux de codage proche de la capacité, le taux d'échec de ces algorithmes sera dramatiquement élevé, car la plupart des mots reçus seront à une distance du code presque deux fois supérieure à la capacité de correction.

En effet, puisqu'il y a 2^{n-k} coset leader et que $\sum_{w=0}^{nH_2^{-1}(1-k/n)} \binom{n}{w} \stackrel{\exp}{=} 2^{n-k}$, on montre facilement que si l'on considère la métrique de Hamming, ceux-ci sont presque tous de poids $nH_2^{-1}(1 \Leftrightarrow k/n)(1 \Leftrightarrow o(1))$, de même que presque tous les mots de poids $nH_2^{-1}(1 \Leftrightarrow k/n)$ sont à distance $o(n)$ d'un coset leader. L'ensemble des cosets leader forme donc une sphère presque parfaite centrée en zéro, avec, marginalement, des points plus proches de son centre. Levitin l'a comparé à un «hérisson retourné».

Or si les codes linéaires binaires algébriques, tels que les BCH ou les codes de Goppa, par exemple, permettent de mettre au point des algorithmes efficaces de décodage borné, ils ne présentent en revanche aucun avantage sur les codes linéaires binaires quelconques (aléatoires) pour ce qui est d'un décodage au delà de la capacité de correction.

Comme par ailleurs, les codes linéaires binaires algébriques et d'une manière générale, presque tous les codes ayant une faible complexité de Kolmogorov, ont de mauvaises propriétés spectrales (à taux de codage k/n «constant», la distance relative, ratio de la distance minimale et de la longueur, tend généralement vers 0, et le taux d'erreur intrinsèquement indécodable vers 1, lorsque l'on fait croître indéfiniment la longueur n), nous ne nous y intéresserons tout simplement pas, et considèrerons des codes aléatoires aux propriétés spectrales moyennes.

1.2.2 Décodage complet

A la différence du décodage borné, le décodage complet[1] a pour tâche de retrouver le coset leader du mot reçu quel que soit le poids de ce coset leader. Cependant, on sait que ce poids est inférieur au rayon de recouvrement, c.a.d. le petit rayon possible pour des boules centrées sur les éléments d'un même coset et devant recouvrir tout l'espace ambiant. Le décodage complet est donc un décodage borné jusqu'au rayon de recouvrement (le cas marginal de non-unicité du coset-leader peut-être traité comme bon semble... on peut par exemple choisir aléatoirement l'un d'entre eux).

Le décodage complet est aussi parfois appelé « décodage à distance minimale » (non pas que l'on décode jusqu'à la distance minimale, mais l'on cherche le mot de code à plus petite distance du mot reçu). Lorsque la métrique considérée est telle que minimiser le poids de l'erreur équivaut à maximiser sa vraisemblance (c'est le cas de la métrique de Hamming lorsque le canal est sans mémoire, binaire et symétrique), on parle alors de « décodage à maximum de

1. Termes anglo-saxons: Complete decoding, minimal distance decoding, maximum likelihood decoding

vraisemblance ». Pour un code donné, ce dernier permet d'obtenir les plus faibles taux d'erreur de décodage qu'un algorithme puisse atteindre.

Nous nous intéressons aux codes généraux, c.a.d. à tous les codes linéaires binaires et non pas seulement à ceux qui appartiennent à une famille de codes dont la description peut être plus concise que la simple donnée de l'une de leur matrice génératrice ou de parité.

On sait depuis 1987 [Bli87] que le rayon de recouvrement de « presque tous » ces codes est proche de la borne de Goblick (initialement, il s'agissait d'une borne inférieure):

Théorème 5 *Le rayon de recouvrement (relativement à la métrique de Hamming) de presque tous les* $[n,k]$*codes linéaires binaires est égal à* $nH_2^{-1}(1 \Leftrightarrow k/n)(1+o(1))$.

(H_2^{-1} étant l'inverse de la fonction entropie binaire $H_2(x) = \Leftrightarrow x \log_2 x \Leftrightarrow (1 \Leftrightarrow x)\log_2(1 \Leftrightarrow x)$ restreinte à la première moitié de l'intervalle $[0,1]$)

Le décodage dur complet peut donc être vu comme un décodage borné jusqu'à une distance légèrement supérieure à $nH_2^{-1}(1 \Leftrightarrow k/n)$.

Il est intéressant (ou rassurant) de noter ici comment la combinatoire et la théorie de l'information se superposent: la capacité d'un canal binaire symétrique de probabilité de transition τ est $C = 1 \Leftrightarrow H_2(\tau)$; le rayon de recouvrement d'un code de longueur n et de taux égal à cette capacité est $nH_2^{-1}(1 \Leftrightarrow C) = n\tau$. Le rayon de recouvrement d'un code de taux maximal est donc égal au poids moyen d'une erreur de transmission (la distance de Hamming moyenne entre mot émis et mot reçu).

1.2.3 Décodage quasi-complet

Alors que la définition du décodage complet s'applique à des algorithmes et des objets combinatoires et ne requiert pas la définition d'un « canal », il n'en va pas de même pour celle du décodage quasi-complet[1].

Dans un contexte de transmission et avec la donnée d'un canal imparfait (bruité), on appelle décodage quasi-complet tout algorithme sous-optimal de décodage tel que la probabilité de mauvais décodage (le coset renvoyé n'est pas le coset leader) soit du même ordre ou plus petite que la probabilité P_{ML} pour que le motif d'erreur soit « intrinsèquement indécodable ».

Par exemple, si un algorithme énumère les N motifs d'erreur les plus vraisemblables de l'espace ambiant, recherchant un motif d'erreur appartenant au coset du mot reçu, le premier trouvé sera le coset leader. Il y aura mauvais décodage si et seulement si aucun de ces N motifs n'est dans le coset du mot reçu. Dumer a montré [Dum96b] que la probabilité de cet évènement est majorée par $\left(1 + \frac{2^{n-k}}{N}\right) P_{ML}$. Si $N = O(2^{n-k})$, le décodage est donc quasi-complet.

La justification de cette contrainte est bien entendu que les taux d'erreur résiduels des décodages complet et quasi-complet soient également du même

1. Termes anglo-saxons: Quasi-complete decoding, near-maximum-likelihood decoding

ordre; et l'intérêt du décodage quasi-complet réside dans le fait que, si l'on tolère cette sous-optimalité (fût-elle négligeable), les coût de calcul qui sont en jeu peuvent s'avérer nettement plus intéressant que ceux d'un décodage optimal, une recherche exhaustive n'étant pas nécessaire.

Pour qu'un algorithme de décodage permette d'atteindre un taux d'erreur résiduel presque optimal, il est nécessaire qu'il soit en mesure de trouver le coset leader du mot reçu avec une grande probabilité, quel que soit ce mot, et il faut alors s'attendre à ce que le coset leader soit de poids proche de $nH_2^{-1}(1 \Leftrightarrow k/n)$.

1.2.4 Critère d'arrêt

Lorsque les algorithmes de décodage complet ou quasi-complet trouvent le coset leader du mot reçu, ils n'ont généralement pas la possibilité de s'en rendre compte immédiatement et poursuivent les calculs (le coset leader restera en mémoire en tant que meilleur candidat jusqu'à la fin des calculs).

Il est cependant parfois possible de s'en rendre compte. Par exemple, pour le décodage dur, lorsqu'un élément du coset est de poids inférieur au rayon d'empilement du code, on peut être assuré de la minimalité de son poids au sein de son coset. Ceci pourra donc constituer un critère d'arrêt, et permettra parfois d'arrêter les calculs de manière précoce, diminuant le coût moyen du décodage. Un critère d'arrêt équivalent existe pour le décodage souple et est décrit en section 3.2.1.

Les critères d'arrêt habituellement utilisés sont infaillibles en ce sens que l'on peut être sûr, si le critère est vérifié, que l'on a affaire au coset leader.

Mais si l'infaillibilité d'un critère est nécessaire pour le décodage à maximum de vraisemblance, il n'en va pas de même pour le décodage quasi-complet, pour lequel il est juste requis que la probabilité pour que l'arrêt soit provoqué par un élément du coset, différent du motif d'erreur, soit du même ordre ou plus petite que P_{ML}.

Par ailleurs, il est souhaitable que la probabilité pour que le motif d'erreur satisfasse le critère soit la plus grande possible (ainsi le critère s'appliquera plus souvent et le nombre moyen d'itérations sera moindre). Cela ne va pas sans rappeler les concepts de probabilité de fausse alarme et de détection que nous étudierons dans la partie «Détection et reconnaissance».

Bien qu'il paraisse envisageable, donc, de définir un critère capturant plus de motifs que les critères infaillibles, au prix d'une augmentation négligeable du taux d'erreur, il n'y a pas eu à ma connaissance de publication abordant ce sujet. Un tel résultat serait pourtant important car utilisé systématiquement ensuite dans tout travail sur le décodage quasi-optimal.

Mes recherches personnelles sur ce sujet sont encore en cours, et n'ont pas abouti à ce jour à un critére plus performant que ceux existants, malgré quelques faux espoirs qui se sont en fait avérés trompeurs. Je préfère donc ne pas encombrer ce manuscrit des formules sous-jacentes à une telle étude.

1.3 Ordre exponentiel

1.3.1 Définitions et propriétés

Les quantités qui seront considérées, seront souvent exponentielles en n, la longueur du code. Ainsi, en va-t-il du taux d'erreur binaire (cf. théorème 4, p. 22), de la complexité d'un algorithme de décodage complet ou quasi-complet, du nombre de mot (mots de code, mots du coset, motifs d'erreur...) dans une boule de Hamming de rayon proportionnel à n, et de bien d'autres acteurs jouant sur la scène de l'espace ambiant $\mathbf{F}_2^n$. Les probabilités seront bien souvent des fonctions exponentiellement décroissantes et les dénombrement des fonctions exponentiellement croissantes.

Par ailleurs, que ce soit pour des raisons de concision ou d'imperfection des modèles, ces quantités seront souvent approximées. Il apparaît donc intéressant de se fixer un degré maximal d'imprécision, et un cadre axiomatique consistant où les approximations ne dépassant pas ce degré s'expriment, s'évaluent, se multiplient ou s'additionnent de manière simple et transparente.

Toute considération quantitative se référant en général au nombre n, il est en effet possible et recommandé d'exprimer de manière simple une vaste classe de résultats. Par exemple l'expression « Telle propriété est vérifiée par presque tous les éléments de $X(n)$ », est parfaitement consistante (le mot « presque » ne signifie pourtant rien de précis dans le langage courant) si on lui attribue la signification « la probabilité pour qu'un élément de $X(n)$ ne satisfasse pas telle propriété tend vers 0 lorsque n tend vers l'infini ».

Si une quantité $Q(n) > 0$ est telle que $\log_2(Q(n))/n$ a une limite α (éventuellement égale à 0) en l'infini, la notation $Q(n) = 2^{\alpha n + o(n)}$, répond de manière satisfaisante à la rigueur et à l'intuition, mais nous allons en définir une autre, plus concise et plus pratique.

Définition 6 *On dit d'une quantité $Q > 0$ dépendant de n qu'elle est d'ordre exponentiel α si :*

$$\lim_{n\to\infty} \log_2(Q(n))/n = \alpha$$

Si deux quantités Q et Q' ont le même ordre exponentiel, on note $Q \stackrel{\exp}{=} Q'$. En particulier si Q est d'ordre exponentiel α, on note $Q(n) \stackrel{\exp}{=} 2^{\alpha n}$.

Si l'ordre exponentiel de Q est strictement inférieur à celui de Q', on note $Q \stackrel{\exp}{<} Q'$. On définit de même, de manière naturelle $\stackrel{\exp}{\leq}$, $\stackrel{\exp}{>}$ et $\stackrel{\exp}{\geq}$.

Les propriétés suivantes de l'ordre exponentiel se vérifient immédiatement :

Proposition 3 *Si deux quantités Q et Q' sont d'ordre exponentiel respectif α et β alors*

- *Quel que soit x réel, Q^x est d'ordre exponentiel αx ;*
- $Q + Q' \stackrel{\exp}{=} 2^{\max(\alpha,\beta)n}$;
- $QQ' \stackrel{\exp}{=} 2^{\alpha+\beta}$.

Toute quantité sous-exponentielle en n (donc tout polynôme) est d'ordre exponentiel 0. Pour tout $x \in R$, toute quantité $Q > 0$ dépendant de n vérifie $Q(\log Q)^x \stackrel{\exp}{=} Q$.

Se fier à l'ordre exponentiel d'une quantité pour l'estimer n'est cependant pas toujours pertinent. Ainsi, si $Q \stackrel{\exp}{>} Q'$, cela signifie certes que pour n suffisamment grand Q est strictement supérieur à Q', mais n'empêche pas que pour un n donné (fini) ce soit le contraire. L'ordre exponentiel n'a donc d'intérêt que dans le cadre des études asymptotiques, qui quant à elles n'en manquent pas et seront menées de manière préliminaire à toute étude plus précise.

Voici deux formules classiques et fort pratiques exploitant la notion d'ordre exponentiel. Elles sont basées sur la formule de Stirling d'approximation de $n!$. On en trouvera les démonstrations dans [Gal68] (p. 530) ou [MS77] (p. 309).

Définition 7 *On appelle entropie binaire la fonction:*

$$H_2(x) = \Leftrightarrow x \log_2(x) \Leftrightarrow (1 \Leftrightarrow x) \log_2(1 \Leftrightarrow x)$$

Proposition 4

$$\forall \alpha \in]0,1[\quad \binom{n}{\alpha n} \stackrel{\exp}{=} 2^{nH_2(\alpha)}$$

$$\forall \alpha \in]0,1/2] \quad \sum_{i=0}^{\alpha n} \binom{n}{i} \stackrel{\exp}{=} 2^{nH_2(\alpha)}$$

1.3.2 Probabilité de succès, quasi-complétude, nombre d'itérations

L'ordre exponentiel est particulièrement bien adapté à l'étude de certains algorithme de décodage quasi-complet.

Supposons en effet qu'un algorithme $A(N)$ ait la forme suivante:

Répéter N fois:
- Choisir aléatoirement un nouveau paramètre;
- Calculer les candidats associés à ce paramètre ainsi que leur vraisemblance
- Garder en mémoire le plus vraisemblable.

N est appelé le nombre d'itérations. Il importe, pour que l'étude qui suit soit valable, que les N itérations soient parfaitement identiques (donc que les différents choix possibles pour le paramètre soient toujours les mêmes) et indépendantes les unes des autres.

Il est alors possible de définir la probabilité de succès d'une itération, que nous noterons $p_{succ}(A)$ ou simplement p_{succ} en l'absence d'ambiguïté, qui est la probabilité pour que le candidat optimal (le coset leader) apparaissent dans l'ensemble des candidats calculés lors d'une itération, et qui est donc la même à chaque itération.

La probabilité d'échec de l'algorithme (les N itérations n'ont pas permis de trouver le candidat le plus vraisemblable) vaut alors $(1 \Leftrightarrow p_{succ})^N$. Le décodage est quasi-complet si le ratio de cette probabilité par la probabilité d'erreur intrinsèquement indécodable P_{ML} reste majoré lorsque n augmente.

Proposition 5 *Pour que le décodage soit quasi-complet,* $N \overset{\exp}{\geq} \frac{1}{p_{succ}}$ *est nécessaire, et* $N = \frac{n \log n}{p_{succ}} \overset{\exp}{=} \frac{1}{p_{succ}}$ *est suffisant.*

Démonstration: Montrons d'abord la condition nécessaire:

Supposons $N \overset{\exp}{<} \frac{1}{p_{succ}}$. Alors $Np_{succ} \overset{\exp}{<} 1$, donc $Np_{succ} \underset{n \to \infty}{\Leftrightarrow} 0$, et en particulier, pour n suffisamment grand $p_{succ} < 1/2$.

La probabilité d'échec de l'algorithme $(1 \Leftrightarrow p_{succ})^N$ vaut aussi:

$$\left((1 \Leftrightarrow p_{succ})^{1/p_{succ}}\right)^{Np_{succ}}$$

Mais $(0 < p_{succ} < 1/2) \Rightarrow 1/4 < (1 \Leftrightarrow p_{succ})^{1/p_{succ}} < e^{-1}$, et puisque Np_{succ} tend vers 0, la probabilité d'échec de l'algorithme tend vers 1. P_{ML} tendant vers 0, le décodage n'est pas quasi-complet.

Passons à la condition suffisante: Supposons $N = \frac{n \log n}{p_{succ}}$. Alors la probabilité d'échec de l'algorithme $(1 \Leftrightarrow p_{succ})^N$ vaut aussi:

$$\left((1 \Leftrightarrow p_{succ})^{1/p_{succ}}\right)^{n \log n}$$

Mais $(p_{succ} > 0) \Rightarrow (1 \Leftrightarrow p_{succ})^{1/p_{succ}} < e^{-1}$. La probabilité d'échec de l'algorithme est donc inférieure à $e^{-n \log n}$. Pour n suffisamment grand, elle est donc inférieure à P_{ML} (cf. théorème 4, p. 22), et le décodage est donc bien quasi-complet.

□

L'ordre exponentiel du nombre d'itération nécessaire pour que le décodage soit quasi-complet est donc l'opposé de celui de la probabilité de succès. Ce résultat très pratique sera utilisé abondamment dans l'étude asymptotique des algorithme de décodage à base d'ensembles d'information (cf. section 2.2, p 46).

Conclusion

Les principales définitions dont nous aurons besoin pour traiter du décodage ont été données.

Nous avons vu que, pour une utilisation optimale du canal, la plupart des motifs d'erreur seront de poids supérieur au rayon d'empilement d'un code aux propriétés spectrales moyennes. Dans ces conditions, les familles de codes linéaires binaires usuellement étudiés, tels les BCH par exemple, sont inintéressantes pour deux raisons: la première est que leur propriétés spectrales sont

moins bonnes que la moyenne, la deuxième est que l'on n'a pas d'algorithme de décodage complet pour ces codes beaucoup plus efficace que ceux s'appliquant à des codes quelconques aux propriétés moyennes.

Nous allons étudier ces algorithmes dans les chapitres suivants, leur complexité sera exponentielle en n. Nous avons vu que les taux d'erreur après décodage sont eux-mêmes exponentiels en n. Nous nous sommes donc fixés un degré maximal d'imprécision pour les futures études quantitatives: en première approximation, les termes exponentiels seront donnés exactement, les autres seront négligés.

Chapitre 2

Décodage dur

L'obscurité est le royaume de l'erreur.
Marquis de Vauvenargues (Réflexions et maximes)

Les plus courtes erreurs sont toujours les meilleures. *Molière (L'Étourdi)*

Une erreur ne devient une faute que lorsqu'on ne veut pas en démordre. *Ernst Jünger*

Introduction

Nous allons donner dans ce chapitre une sélection des algorithmes les plus classiques de décodage dur complet ou quasi-complet des codes linéaires binaires généraux et leur complexité moyenne, du plus trivial au plus rapide. Nous n'écrirons rien sur le décodage par treillis [LV95] [ZS93], car malgré d'intéressants avantages (en particulier pour le décodage souple) sa complexité reste redhibitoire. Nous ne parlerons pas non plus des techniques de gradient de type « zero-neighbor algorithm » [Hwa79] [LH85] [Han98], qui nécessitent de lourds calculs préliminaires pour chaque nouveau code et dont la complexité hors calculs préliminaires est de toute façon supérieure à celle des algorithmes que nous allons étudier.

Dans ce chapitre, le canal binaire symétrique et la métrique de Hamming sont considérés. Le décodage complet consiste à trouver le coset leader du mot reçu, ou de manière équivalente, le mot de code le plus proche du mot reçu. Nous supposerons toujours qu'il est unique, le cas contraire étant très marginal et son traitement (prise de décision, premier trouvé ou pseudo-aléa par exemple) ayant une influence de toute façon négligeable sur les taux d'erreur et les complexités.

Lorsque le poids du coset leader est inférieur au rayon d'empilement (ou capacité de correction) du code, les algorithmes peuvent s'arrêter sitôt qu'ils l'ont trouvé car on sait qu'il ne peut y avoir de coset de poids inférieur. Dans ce

cas, la compléxité du décodage est fortement réduite. Dans le cas contraire en revanche, rien ne permet en général, si ce n'est la comparaison à tous les autres éléments du coset, d'être sûr que l'élément du coset trouvé soit le coset leader. Le décodage complet requiert donc une certaine exhaustivité, contrairement au décodage quasi-complet, souvent probabilste. Les complexités mentionnées correspondent au cas où le coset leader est de poids $w = nH_2^{-1}(1 \Leftrightarrow k/n)$ (approximation du rayon de recouvrement, ou poids moyen des erreurs pour un taux de codage égal à la capacité), c.a.d. au pire cas.

La complexité des algorithmes que nous allons considérer peut toujours s'exprimer sous la forme $2^{ne(R)}P(n)$ où R est le taux de codage k/n, e une fonction à valeurs réelles positives et P un polynôme. Nous ne nous intéresserons en général qu'à l'ordre exponentiel des complexités: $2^{ne(R)}P(n) \stackrel{\exp}{=} 2^{ne(R)}$. La quantité $e(R)$ sera appelée coefficient de complexité.

Par exemple, dans le cas d'un algorithme de décodage trivial qui calcule la distance du mot reçu à chaque mot de code, on a $e(R) = R$. En effet, si le calcul de la distance entre deux mots coûte $K_d n$ (K_d étant une constante), le coût total de l'algorithme est $2^k K_d n \stackrel{\exp}{=} 2^{nR}$ (car il y a 2^k mots de code). Le coefficient de complexité est donc dans ce cas égal au taux de codage.

Nous allons voir dans quelle mesure, historiquement, on a pu faire baisser progressivement le coefficient de complexité.

2.1 Décodage par recherche du motif d'erreur

Soit H une $(n \Leftrightarrow k) \times n$-matrice de parité du code, pour tout mot m de l'espace ambiant, on appelle syndrôme de m le vecteur colonne Hm^t (il vérifie une propriété de linéarité: $H(m + m')^t = Hm^t + Hm'^t$).

Le mot émis ayant un syndrôme nul, l'erreur de transmission a le même syndrôme que le mot reçu. Décoder revient donc a chercher le mot de plus petit poids ayant le même syndrôme que le mot reçu.

Une première méthode est de stocker préalablement dans une table, pour chaque syndrôme, le mot en question. La complexité en temps, hors calculs préliminaires, est alors pour ainsi dire nulle, mais la complexité en espace est de $2^{n(1-R)}$ et les calculs préliminaires sont à refaire pour chaque nouveau code.

Cela peut aussi se faire en calculant le syndrôme de tous les mots de l'espace ambiant que l'on génère par poids croissant jusqu'à ce que l'un d'eux correspondent à celui du mot reçu.

Nous allons étudier cet algorithme, non pas tant pour lui même que pour les trois améliorations significatives que l'on peut, nous allons le voir, lui apporter. Celles-ci pourront en effet introduire quelques idées dont nous nous servirons par la suite pour améliorer les algorithmes de décodage par ensemble d'information.

Un travail clair, précis et complet a été mené par Ilya Dumer sur ces trois améliorations [Dum89], [Dum93], [Dum96b], [Dum99], [Dum01]. Le dernier article, traitant de la dernière amélioration, devrait paraître en 2001.

2.1.1 Complexité

L'algorithme génère tous les motifs d'erreur par poids croissant et teste si le mot reçu auquel on retranche le motif d'erreur appartient au code (teste si le mot reçu et le motif d'erreur ont même syndrôme). Il renvoit le premier ayant cette propriété.

Il y a $\binom{n}{i}$ motifs d'erreur de poids i. Si le coset leader est de poids $w = nH_2^{-1}(1 \Leftrightarrow R)$, l'algorithme génèrera $\sum_{i=0}^{w} \binom{n}{i} \stackrel{\exp}{=} 2^{nH_2(w/n)} \stackrel{\exp}{=} 2^{n(1-R)}$ motifs distincts, le traitement de chacun d'entre eux se faisant en temps polynomial (calcul d'un syndrôme, test d'égalité).

Le rayon de recouvrement peut être plus grand que $nH_2^{-1}(1 \Leftrightarrow R)$ il ne s'agit pas alors de décodage complet, mais Evseev a montré [Evs83] que la probabilité d'erreur de tout algorithme de décodage explorant les $2^{n(1-R)}$ motifs d'erreur de plus petit poids était majoré par deux fois la probabilité P_{ML} pour que le motif d'erreur soit « intrinsèquement indécodable »; Dumer a quant à lui montré [Dum96b] que s'il explore les N motifs d'erreur de plus petit poids, quel que soit N, la probabilité d'erreur est majorée par $\left(1 + \frac{2^{n(1-R)}}{N}\right) P_{ML}$. On a donc au pire un décodage quasi-complet.

Le coefficient de complexité est donc, c'est notoire[1], $1 \Leftrightarrow R$.

2.1.2 Restriction à un ensemble d'information

Appelons $(1...n)$ le support de l'espace ambiant, et supposons, sans perte de généralité et quitte à s'en remettre à une permutation de ce support, que $(1...k)$ représente un ensemble d'information, c.a.d. un sous-ensemble du support de cardinalité égale à la dimension du code et tel que la sous-matrice extraite d'une quelconque matrice génératrice du code en ne gardant que les colonnes indéxées par cet ensemble soit de rang plein.

Lorsqu'il génère un nouveau motif d'erreur, l'algorithme précédent déterminait tous les bits de ce motif. L'amélioration consiste ici à ne déterminer que les k premiers bits et à compléter ce début de motif de manière à ce que le motif obtenu ait le même syndrôme que le mot reçu.

Pour ce faire, on a juste besoin d'une matrice systématique à gauche du code (rappelons que $(1...k)$ est un ensemble d'information). En effet, en réencodant par cette matrice le mot reçu restreint aux k premières positions, auquel on retranche le début de motif, on obtient un mot de code qui ne diffère du mot reçu sur les k premières positions que par le début de motif considéré. Si on le retranche au mot reçu, on obtient le seul motif d'erreur égal à notre début de motif sur l'ensemble d'information et de même syndrôme que le mot reçu. Bien sûr, le complément ainsi calculé pour le motif ne sera généralement pas de poids faible, mais ce sera le cas si les k premiers bits correspondent à ceux du coset leader.

1. Cette complexité est généralement mentionnée comme étant celle du décodage par treillis. Mais si les versions basiques de ces deux algorithmes se ressemblent en ce qu'ils testent plus ou moins les mêmes candidats (mais les algorithmes et l'ordre des tests sont différents), leurs versions améliorées n'ont, quant à elles, rien à voir.

Pour trouver le coset leader, l'algorithme précédent doit avoir juste sur les n décisions prises pour déterminer le motif, tandis qu'avec l'amélioration il suffit qu'il ait juste sur les k premières.

Une première version de cet algorithme consiste alors à générer tous les motifs de poids inférieur ou ègal à w sur la fenêtre d'information, on obtient alors une complexité égale à $2^{o(n)} \sum_{i=0}^{w} \binom{k}{i} \stackrel{\exp}{=} 2^{nRH_2(\frac{w}{nR})}$ qui est bien-sûr inférieure à $2^{o(n)} \sum_{i=0}^{w} \binom{n}{i}$, la complexité de l'algorithme sans amélioration.

Une deuxiéme possibilité [Evs83], qui s'avère plus économique, est de considérer plusieurs ensembles d'information de manière à être assuré que sur au moins l'un d'entre eux le poids de l'erreur est au plus wR (pour un code cyclique on peut par exemple prendre l'ensemble $(1...k)$ et tous ses shifts successifs. D'une manière générale, pour presque tous les codes, moins de n ensembles d'information suffisent). On ne génère donc les différents motifs que jusqu'au poids wR au lieu de w.

La complexité est alors de $2^{o(n)} \sum_{i=0}^{wR} \binom{k}{i} \stackrel{\exp}{=} 2^{nRH_2(\frac{wR}{nR})}$.

Si $w = nH_2^{-1}(1 \Leftrightarrow R)$, elle est donc d'ordre exponentiel $2^{nR(1-R)}$.

Le nouveau coefficient de complexité est donc $R(1 \Leftrightarrow R)$.

2.1.3 Séparation en deux des mots de l'espace ambiant

L'algorithme de base ne sollicite quasiment pas la mémoire de l'ordinateur. Cette situation indique souvent la possibilité d'un meilleur compromis temps/mémoire. Nous allons voir comment la complexité de l'algorithme de base peut être réduite à sa racine carrée, ou comment diviser par deux le coefficient de complexité $1 \Leftrightarrow R$.

Pour tout mot m de l'espace ambiant, soit $s(m) = Hm^t$ son syndrôme (H est une matrice de parité donnée du code). Soient $I_1 = (1...n/2)$ et $I_2 = (n/2+1...n)$ partitionnant le support en deux parties de même cardinalité, et soient H_1, H_2, m_1 et m_2 les restrictions de la matrice H et du mot m aux ensembles I_1 et I_2 respectivement.

On vérifie immédiatement que $Hm^t = H_1 m_1^t + H_2 m_2^t$, notons alors $s_1(m_1) = H_1 m_1^t$ et $s_2(m_2) = H_2 m_2^t$ ce que nous appellerons les syndrômes partiels.

Rechercher un mot m de longueur n, de syndrôme donné, disons égal à $\tilde{s}$, et de poids minimal revient donc à chercher deux mots m_1 et m_2 de longueur $n/2$, tels que $s_1(m_1) + s_2(m_2)$ soit égal $\tilde{s}$ et $\mathrm{wt}(m_1) + \mathrm{wt}(m_2)$ soit minimal.

Nous allons donc générer par poids croissant, et jusqu'à un certain poids p, des mots m_1 et m_2 de longueur $n/2$ et les stocker, ainsi que leur syndrôme partiel dans des tables T_1 et T_2.

Il suffit ensuite de chercher un couple (m_1, m_2), tel que $s_1(m_1) + s_2(m_2)$ soit égal $\tilde{s}$ (et l'on a bien sûr $\mathrm{wt}(m_1) + \mathrm{wt}(m_2) \leq 2p$).

Algorithmiquement parlant, deux solutions se présentent pour avoir accès à ces couples:

- ou bien grâce à un tri préliminaire des ensembles de couples $(m, s_i(m))$ selon un ordre sur les $s_i(m) \in \mathbf{F}_2^{n-k}$; la complexité d'un tri a le même ordre exponentiel que le cardinal de l'ensemble et n'augmentera donc pas

l'ordre exponentiel de la complexité de l'algorithme qui est minorée par ce cardinal (il n'empêche que, bien que de même ordre exponentel, la complexité de ce tri dominera celle de la première étape par un facteur polynomial en n).

– Ou bien grâce à un rangement au fur et à mesure des m dans des boîtes correspondant à la valeur des $s_i(m)$; la complexité en espace (nombre de boîtes) devient alors d'ordre exponentiel $n \Leftrightarrow k$; lequel se répercute sur la complexité en temps pour des raisons de gestion de la mémoire (coût de l'allocation, de l'initialisation, de la libération des boîtes).

La deuxième solution est ici exclue car la complexité de l'algorithme hors gestion de la mémoire est, nous allons le voir, d'ordre exponentiel strictement inférieure à $n \Leftrightarrow k$; et utiliser la deuxième solution annulerait cette amélioration. On adoptera donc la solution du tri.

Les coûts en temps et en mémoire de ces différentes étapes, si elles sont correctement implémentées, sont du même ordre exponentiel que le nombre de mots ainsi générés.

Que doit valoir p pour que l'algorithme donne la solution ? L'ideal serait d'avoir $p = w/2$ pour avoir le plus faible coût. Mais l'on n'est pas sûr que le coset leader se sépare sur (I_1, I_2) en deux parties de même poids. Par ailleurs en prenant p plus grand, on ne peut être assuré que la première solution trouvée soit la meilleure, ce qui complique la deuxième phase...

La solution consiste en fait à considérer une fois encore plusieurs partitions (I_1, I_2) différentes. On prendra les $n/2$ premiers shifts de la partition (I_1, I_2) définies plus haut. On vérifie en effet aisément que le poids de tout mot de poids w se décompose en $(w/2, w/2)$ sur l'une au moins de ces $n/2$ partitions. On peut alors ne calculer que les mots m_1 et m_2 de poids $w/2$, s'il existe un élément du coset de poids w, il sera ainsi détecté.

Quelques adaptations simples sont à faire dans le cas où n et/ou w sont impairs mais ne posent pas vraiment de problèmes [Dum99].

Il faut générer et stocker en mémoire $n\binom{n/2}{w/2} \overset{\exp}{=} 2^{\frac{n}{2}H_2(w/n)}$ mots, donc $2^{n/2(1-R)}$ si $w = nH_2^{-1}(1 \Leftrightarrow R)$.

Le nouveau coefficient de complexité vaut donc $\frac{1-R}{2}$.

2.1.4 Utilisation de codes poinçonnés

On voudrait utiliser simultanément les deux améliorations précédentes et obtenir ainsi un coefficient de complexité de $\frac{R(1-R)}{2}$, mais cela n'est pas possible car si l'on connaît la valeur du syndrôme de l'erreur sur le support complet, on n'a en revanche aucune information sur la valeur du syndrôme de l'erreur restreinte à un ensemble d'information.

Il est cependant possible de connaître la valeur du syndrôme de l'erreur restreinte à un sous-ensemble du support relativement à la matrice de parité du code poinçonné en le complémentaire de ce sous-ensemble. On pourra, grâce à

cette astuce, regrouper indirectement ces deux améliorations en un seul algorithme. Le coefficient de complexité ne sera pas $\frac{R(1-R)}{2}$ mais $\frac{R(1-R)}{1+R}$. Voici le principe:

Soit s ($k < s < n$) un paramètre de l'algorithme. Supposons, par souci de simplicité, que s et w soient pairs. Et soient I_1 et I_2 deux sous-ensembles disjoints du support, de cardinalité $s/2$ tels que $I = I_1 \cup I_2$ contienne un ensemble d'information.

Soit C_I le $[s,k]$-code poinçonné, c'est-à-dire l'ensemble des restrictions des mots de code à l'ensemble I; et soit H_I une $(s \Leftrightarrow k) \times s$-matrice de parité de ce code.

Soit m le mot reçu, m_I sa restriction à I, et $\tilde{s}$ le syndrôme selon H_I de m_I.

Soient enfin H_{I_1} et H_{I_2} les restrictions de H_I aux ensembles I_1 et I_2 et pour tous mots m_1 et m_2 de longueur $s/2$, $s_1(m_1) = H_{I_1} m_1^t$ et $s_2(m_2) = H_{I_2} m_2^t$ leurs syndrômes partiels.

Le fait que I contienne un ensemble d'information du code implique que tout mot de code poinçonné c_I ne peut être étendu qu'en un seul mot de code c, que l'on notera $C(c_I)$, et que cela peut se faire par simple réencodage de la partie « information » de c_I.

Nous allons supposer que I_1 et I_2 contiennent chacun au plus $w\frac{s}{2n}$ erreurs de transmission. L'hypothèse est abusive, mais comme dans les deux cas précédents, on considèrera plusieurs couples (I_1, I_2) choisis de manière à ce que l'on soit sûr qu'au moins l'un d'entre eux la vérifie. Dumer a montré, de manière constructive, que $O(n)$ couples (I_1, I_2) suffisent [Dum96b] (voir aussi [Bar98]).

On cherche alors deux mots m_1 et m_2 de longueur $s/2$, de poids inférieur ou égal à $w\frac{s}{2n}$, tels que $s_1(m_1) + s_2(m_2) = \tilde{s}$, et tels que $m + C(m_I + (m_1|m_2))$ soit de poids minimum.

Nous allons construire tous les mots de longueur $s/2$ et de poids inférieur ou égal à $w\frac{s}{2n}$. Et pour chaque tel mot e, nous stocquerons les couples $(e, s_1(e))$ et $(e, s_2(e))$. Cette opération aura un coût d'ordre exponentiel $2^{\frac{s}{2}H_2(\frac{w}{n})}$.

Nous allons ensuite considérer les paires (e_1, e_2) telles que $s_1(e_1)+s_2(e_2) = \tilde{s}$. On montre qu'en moyenne une paire sur 2^{s-k} vérifie cette égalité. Pour chaque telle paire, on calcule le poids de $m + C(m_I + (e_1|e_2))$ et si ce poids est le plus petit que l'on ait trouvé, on garde en mémoire le mot de code $C(m_I + (e_1|e_2))$. Cette opération aura un coût moyen d'ordre exponentiel $2^{sH_2(\frac{w}{n})-(s-k)}$.

Algorithmiquement parlant, deux solutions se présentent pour avoir accès à ces paires:

- ou bien grâce à un tri préliminaire des ensembles de couples $(e, s_i(e))$ selon un ordre sur les $s_i(e) \in \mathbf{F}_2^{s-k}$; la complexité d'un tri a le même ordre exponentiel que le cardinal de l'ensemble et n'augmentera donc pas l'ordre exponentiel de la complexité de l'algorithme qui est minorée par ce cardinal (il n'empêche que, bien que de même ordre exponentel, la complexité de ce tri dominera celle de la première étape par un facteur polynomial en n).

- Ou bien grâce à un rangement au fur et à mesure des e dans des boîtes

correspondant à la valeur des $s_i(e)$; la complexité en espace (nombre de boîtes) devient alors d'ordre exponentiel $s \Leftrightarrow k$; lequel se répercute sur la complexité en temps pour des raisons de gestion de la mémoire (coût de l'allocation, de l'initialisation, de la libération des boîtes).

Si la complexité de l'algorithme hors gestion de la mémoire est d'ordre exponentiel supérieur ou égal à $s \Leftrightarrow k$, et si les ressources en mémoire le permettent, la deuxième solution sera préférée car elle permet de faire l'économie du tri.

A la fin de ces opérations, le mot de code gardé en mémoire est celui que l'on recherchait. Le coût de l'algorithme, pour $w = nH_2^{-1}(1 \Leftrightarrow R)$, est d'ordre exponentiel $2^{\max\left(\frac{s}{2}(1-R), s(1-R)-(s-k)\right)}$. Le coefficient de complexité est donc égal à $\max\left(\frac{s}{2}(1 \Leftrightarrow R), s(1 \Leftrightarrow R) \Leftrightarrow (s \Leftrightarrow k)\right)/n$.

Le minimum est atteint pour $s = n\frac{2R}{1+R}$, on a alors:

$$\frac{s}{2}(1 \Leftrightarrow R) = s(1 \Leftrightarrow R) \Leftrightarrow (s \Leftrightarrow k) = s \Leftrightarrow k = n\frac{R(1 \Leftrightarrow R)}{1+R}$$

Le nouveau coefficient de complexité est donc $\frac{R(1-R)}{1+R}$.

La complexité en espace est du même ordre exponentiel que la complexité en temps ($2^{\frac{s}{2}H_2(\frac{w}{n})} = 2^{n\frac{R(1-R)}{1+R}}$) et l'on remarque en particulier qu'elle est donc d'ordre exponentiel 2^{s-k}. La solution des boîtes plutôt que celle du tri n'augmente donc la complexité en espace que par une constante, et diminue la complexité en temps, du fait de l'économie du tri, par un facteur linéaire en n.

2.2 Décodage par ensemble d'information

2.2.1 Principe et justification

J'appellerai algorithme de *décodage par ensemble d'information* tout algorithme de décodage des codes linéaires généraux (aléatoires) explorant un nombre exponentiel (en n) d'ensembles d'information (contrairement aux algorithmes des sections 2.1.2 et 2.1.4, qui travaillent sur $O(n)$ ensembles d'information différents).

Le décodage par ensemble d'information est un principe puissant, réduisant fortement les complexités usuelles. Introduit trés tôt (1962) pour les codes cycliques [Pra62], généralisé par Omura dans un rapport de 1969 (cf. [CC81] pp.119-127), il a depuis été redécouvert et largement étudié, citons entre autres, par ordre chronologique [McE78], [CG81], [Kro89], [CG90], [Dum91], [vT94], [CC98], [BKvT99].

Pour le comparer à la recherche de motif d'erreur, disons que plutôt que d'avoir besoin de trouver précisément l'erreur, le décodage par ensemble d'information a juste besoin d'en trouver un ($n \Leftrightarrow k$)$\Leftrightarrow$recouvrement, ce qui est nettement plus facile.

Définition 8 *Soient X un ensemble, e un sous-ensemble de X et r un entier. on appelle $r\Leftrightarrow$recouvrement de e tout sous-ensemble de X de cardinalité r contenant e, on dit qu'il recouvre e.*

Si E est un ensemble de sous-ensembles de X, on appelle $r\Leftrightarrow$recouvrement de E tout ensemble R de sous-ensembles de X de cardinalité r, tel que tout élément de E soit recouvert par au moins un élément de R.

Si l'on a un ($n \Leftrightarrow k$)$\Leftrightarrow$recouvrement du motif d'erreur, son complémentaire (de cardinalité k) est dépourvu d'erreur, et pour peu qu'il s'agisse d'un ensemble d'information, il est alors possible d'annuler le coset du mot reçu sur cet ensemble et donc de retrouver l'erreur.

S'il ne s'agit pas d'un ensemble d'information, c.a.d. si la matrice extraite de la matrice génératrice en ne gardant que les colonnes indéxées par cet ensemble est de rang non plein $k \Leftrightarrow x$ ($x > 0$) (On dit alors de cet ensemble qu'il est $x\Leftrightarrow$défectif), alors le motif d'erreur est à chercher parmi au plus 2^x cosets du mot reçu, facilement calculables. Il suffit en effet de compléter l'ensemble par x éléments adéquats de manière à ce qu'il contienne un ensemble d'information, et d'essayer les 2^x motifs possibles sur le complément.

Par ailleurs, on montre que la valeur maximale qui puisse être prise par x est négligeable devant k, ainsi que l'affirme de manière plus rigoureuse le théorème suivant [CG90].

Théorème 6 *Pour tout taux de codage R, tout $\alpha > 0$, et tout n suffisamment grand, presque aucun $[n, \lfloor nR \rfloor]\Leftrightarrow$code linéaire ne contient de $\lfloor nR \rfloor$-uplet $\lfloor nR\alpha \rfloor$-défectif.*

Enfin, les implémentations que nous allons proposer ne testent de toute façon que des k⇔uplets non défectifs. Le lemme suivant, attribué à D. Mandelbaum [Man80] est à ce titre rassurant:

Lemme 1 *Soit w une métrique sur $\mathbf{F}_2^n$ telle que:*

$$\forall x, x' \in \mathbf{F}_2^n \quad (\operatorname{supp} x \subsetneq \operatorname{supp} x') \Rightarrow (w(x) < w(x'))$$

Pour tout code C et tout coset leader m (relativement à la métrique w), il existe au moins un ensemble d'information de C disjoint de m.

Démonstration: Permutons le code de manière à ce que le support de m en constitue les positions les plus à droite. Considérons une matrice génératrice de C réduite à sa forme échelon, et soit c sa dernière ligne.

Soit i la position non nulle la plus à gauche de c, et j celle de m. Si $j > i$ l'ensemble d'information de la forme échelon est disjoint du support de m, et si $j \leq i$, alors c a son support inclus dans celui de m.

Donc $m + c$ a son support strictement inclus dans celui de m et donc $w(m + c) < w(m)$, ce qui est incompatible avec le fait que m est un coset leader. □

2.2.2 Description d'une itération, format des données

Un algorithme de décodage par ensemble d'information consiste à chercher un ensemble d'information dépourvu d'erreur. Si l'on en trouve un, il est alors possible de calculer l'élément du coset n'ayant que des 0 sur cet ensemble, et il s'agit forcément de l'erreur (mais l'on ne sait pas forcément la distinguer d'un autre élément du coset).

Nous supposerons, quitte à permuter les colonnes, que l'ensemble d'information considéré est l'ensemble des k positions les plus à gauche. La matrice génératrice sera toujours sous forme systématique (ce qui demande une diagonalisation à chaque changement d'ensemble d'information), et puisqu'il est loisible de considérer n'importe quel élément du coset du mot reçu m à la place de celui-ci, nous considèrerons que m est égal à l'élément de son coset qui est nul sur l'ensemble d'information.

$$G = \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} = \left[\begin{array}{ccc|c} 1 & & (0) & \\ & \ddots & & G_R \\ (0) & & 1 & \end{array}\right] \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} k \overbrace{\qquad\qquad}^{n \Leftrightarrow k}$$

$$m = \left[\begin{array}{ccc|c} 0 & \ldots & 0 & m_R \end{array}\right]$$

Ainsi, $g_1, ...g_k$ et m ayant toujours la même forme sur les k positions les plus à gauche, il n'est pas nécessaire de stocker en mémoire les $k(k+1)$ bits correspondants. Seuls les $(n \Leftrightarrow k)(k+1)$ bits des $n \Leftrightarrow k$ positions, dites de redondance, les plus à droite (G_R et m_R) seront stockés.

Remarque 1 *Les $k+1$ lignes en question, de longueur $n \Leftrightarrow k$, seront stockées sous forme de mots machines (de la largeur du bus de données). Il sera ainsi posssible par exemple d'effectuer un grand nombre d'addition modulo 2 (ou exclusif) simultanément (32 ou 64 pour la plupart des architectures actuelles).*

L'algorithme calcule le poids de m sur ces $n \Leftrightarrow k$ positions (c.a.d. le poids de m_R). S'il est inférieur au rayon d'empilement du code, il ne peut s'agir que du coset leader. L'algorithme le reconstruit en allouant les k positions effacées et en effectuant la bonne permutation.

Sinon, l'algorithme considère un nouvel ensemble d'information, recalcule, par une éliminination de Gauss ou pivot, G_R puis m_R et recommence.

L'algorithme garde en mémoire le coset de plus petit poids parmi ceux qu'il a calculés. Il renvoie ce dernier si au bout d'un nombre d'itérations N_{max} donné il n'en a pas trouvé de poids inférieur au rayon d'empilement.

Le taux d'erreur résiduel est fonction de N_{max} et décroit asymptotiquement vers le taux d'erreur résiduel du décodage complet lorsque N_{max} tend vers l'infini. La valeur de N_{max} nécessaire pour obtenir un taux d'erreur résiduel inférieur à x fois ce taux ($x > 1$ donné) (c'est à dire pour que le décodage soit quasi-complet) croît exponentiellement avec la longueur quel que soit x.

2.2.3 Pivot vicinal

Lorsque deux ensembles d'information ne diffèrent que par une position, on parle alors d'ensembles d'information « vicinaux », et on apelle pivot « vicinal » le fait d'obtenir une matrice diagonalisée sur l'un à partir d'une matrice diagonalisée sur l'autre.

Il s'agit donc de permuter une position d'information et une position de redondance et d'effectuer le pivot correspondant. Mais de même que tout $k \Leftrightarrow$uplet de positions ne constitue pas forcément un ensemble d'information, toutes les permutations ne permettront pas d'effectuer un pivot.

Nous allons donner une condition nécessaire et suffisante sur les deux positions à permuter pour que le pivot soit possible avant de décrire dans le détail la routine qui l'effectuera.

Proposition 6 *Soit I un ensemble d'information, $\bar{I}$ son complémentaire et G la matrice génératrice diagonalisée selon I. Alors pour tout couple $(\lambda, \mu) \in I \times \bar{I}$, l'ensemble obtenu à partir de I en permutant λ et μ est un ensemble d'information si et seulement si $G_{\lambda,\mu} = 1$.*

Démonstration: Un $k \Leftrightarrow$uplet J est un ensemble d'information si et seulement si les colonnes de G indéxées par J sont linéairement indépendantes.

Les colonnes indéxées par $I \Leftrightarrow \{\lambda\}$ sont nulles sauf en une position, distincte pour chaque colonne. Elles sont donc linéairement indépendantes et il n'y a que la position λ en laquelle elles soient toutes nulles.

il faut et il suffit donc que la colonne μ soit indépendante de ces dernières; ce qui est le cas si et seulement si elle est non nulle en λ. □

Nous avons vu que seule la partie redondante G_R de G est stoquée en mémoire. Mettre à jour G_R en tenant compte d'un pivot selon deux positions λ et μ vérifiant la condition ci-dessus revient donc à effectuer les opérations suivantes:

1 Compléter G_R pour obtenir G (systématique à gauche);

2 permuter les colonnes λ et μ de G;

3 diagonaliser à gauche la nouvelle matrice par des combinaisons linéaires de lignes adéquates;

4 éliminer la partie systématique et stocquer dans G_R la partie redondante obtenue.

En étudiant plus précisément l'étape 3, nous allons voir comment s'affranchir des étapes 1, 2 et 4: au début de l'étape 3, la matrice est presque systématique, seule la colonne λ doit être modifiée par les combinaisons linéaires:

$$
\begin{array}{cc}
 & \begin{array}{ccccccccccc} & & & & \lambda & & & & & \mu & & \end{array} \\
 & \begin{array}{ccccccccccc} & & & & \downarrow & & & & & \downarrow & & \end{array} \\
G = & \left[\begin{array}{ccccccccccc}
1 & & & & ? & & & & & 0 & & \\
 & \ddots & & & \vdots & & & & & \vdots & & \\
 & & \ddots & & ? & & (0) & & & 0 & & \\
 & & & & 1 & & & & (?) & 1 & (?) & \leftarrow \lambda \\
 & (0) & & & ? & \ddots & & & & 0 & & \\
 & & & & \vdots & & \ddots & & & \vdots & & \\
 & & & & ? & & & 1 & & 0 & &
\end{array}\right] \\
m = & \left[\begin{array}{ccccccccccc}
 & (0) & & & ? & & (0) & & & (?) & 0 & (?) &
\end{array}\right]
\end{array}
$$

Il s'agit donc de transformer, par une combinaison linéaire de ligne adéquate, chaque « 1 » de la colonne λ, sauf celui de la diagonale, en « 0 ». Les combinaisons linéaires adéquates en question consistent bien sûr à ajouter la ligne λ aux lignes ayant un « 1 » sur la colonne λ. Ce faisant, on voit que la colonne λ prendra la forme de la colonne μ, ce qui est le but recherché, mais aussi que la colonne μ prendra la forme de la colonne λ. Tout se passe donc, aprés cette étape 3, comme si l'on n'avait pas permuté les deux colonnes à l'étape 2, qui apparaît donc superflue.

Supposons donc que l'on n'opère que sur la matrice G_R, c.a.d. que l'on oublie les étapes 1,2 et 4. On vient de voir que la colonne μ reste inchangée. Quelles

transformations le reste de la matrice G_R doit-il subir? Il faut, comme on vient de le voir, ajouter la ligne λ aux lignes ayant un « 1 » sur la colonne que l'on aurait envoyée en λ à l'étape 2, c.a.d. la colonne μ, à ceci près donc, que cette colonne μ ne doit pas en être affectée. Voici donc la routine que je propose, avec les notations suivantes:

Pour $i = 1...k$, R_i représentera la $i$ème ligne de G_R, et pour $j = k+1...n$, $R_{i,j} = G_{i,j}$. Les additions sont des « ou exclusifs » sur les bits.

Pivot(R, m_R, λ, μ)
Si $R_{\lambda,\mu} = 0$ alors ...ERREUR, STOP...
$R_{\lambda,\mu} \leftarrow 0$ /*pour ne pas affecter la colonne μ */
Pour i allant de 1 à k, $i \neq \lambda$, faire:
  Si $R_{i,\mu} = 1$ alors $R_i \leftarrow R_i + R_\lambda$
Si $m_\mu = 1$ alors $m_R \leftarrow m_R + R_\lambda$
$R_{\lambda,\mu} \leftarrow 1$

Le test à l'intérieur de la boucle sera positif une fois sur 2 en moyenne. La complexité de cette routine est donc bien $\frac{k+1}{2}(n \Leftrightarrow k) \approx \frac{R(1-R)}{2}n^2$.

Bien sûr, il faudra garder en mémoire la permutation globale qu'aura subie la matrice G au fil des itérations afin de pouvoir reconstruire quand on le désire le coset choisi. La routine décrite ci-dessus devra donc en plus mettre à jour cette permutation globale.

2.2.4 Complexité

Coût d'une itération

Le coût de chaque itération est le coût d'un pivot plus celui du calcul du poids de m_R. Si l'on dispose d'une matrice diagonalisée sur l'ensemble d'information I comme point de départ, et que l'on veut obtenir une matrice diagonalisée sur l'ensemble d'information J, le plus économique est d'effectuer $k \Leftrightarrow |I \cap J|$ pivots vicinaux.

Lemme 2 *Si I et J sont deux sous-ensembles de $\{1...n\}$ de cardinal k choisis aléatoirement (de manière uniforme dans l'ensemble des tels sous-ensembles) et de manière indépendante, alors l'espérance du cardinal de $I \cap J$ est égale à $\frac{k^2}{n}$.*

Démonstration: Si I est un sous-ensemble de $\{1...n\}$ de cardinal k choisi aléatoirement de manière uniforme dans l'ensemble des tels sous-ensembles, alors chaque position appartient à I avec une probabilité $\frac{k}{n}$. On a donc:

$$E(|I \cap J|) = \sum_{i=1}^{n} \Pr(i \in I \cap J) = \sum_{i=1}^{n} \Pr(i \in I) \Pr(i \in J) = n \left(\frac{k}{n}\right)^2$$

□

$k \Leftrightarrow |I \cap J|$ vaut donc en moyenne $nR(1 \Leftrightarrow R)$. La complexité moyenne d'un pivot vicinal (avec mise à jour de m_R) vaut quant à elle $\frac{(k+1)(n-k)}{2} \approx n^2 \frac{R(1-R)}{2}$. La complexité moyenne d'un pivot est donc $n^3 R^2 (1 \Leftrightarrow R)^2/2$.

Celle du calcul du poids de m_R, linéaire en n, est donc négligeable, et les itérations de l'algorithme de décodage par ensemble d'information coûte en moyenne $n^3 R^2 (1 \Leftrightarrow R)^2/2$.

Nombre d'itérations

Un algorithme de décodage par ensemble d'information teste successivement différents ensembles d'information en espérant que le complémentaire de l'un d'eux recouvre l'erreur. Nous avons deux possibilités:

Ou bien il choisit aléatoirement un certain nombre N d'ensembles d'informations. On a alors un algorithme probabiliste et si N est correctement dimensionné, il s'agit de décodage quasi-complet (nous allons étudier la dépendance de N en n et k pour que ce soit le cas).

Ou bien il parcourt une liste prédéfinie ou calculable d'ensembles d'information qui est telle que l'ensemble de leurs complémentaires recouvre l'ensemble des cosets leader. On peut alors être assuré que le complémentaire d'au moins l'un d'entre eux recouvre le coset leader du mot reçu et donc que celui-ci sera calculé. Il s'agit alors d'un algorithme déterministe de décodage complet.

Définition 9 *On appelle sphère de Hamming de rayon w sur un ensemble X, l'ensemble de tous les sous-ensembles de X de cardinal w.*

Les cosets leader étant de poids au plus le rayon de recouvrement du code, il suffit d'avoir une liste d'ensembles d'information telle que l'ensemble de leurs complémentaires recouvre la sphère de Hamming de rayon le rayon de recouvrement du code.

Quel que soit le code considéré, on sait qu'une telle liste existe. On connait même [ES74] des bornes inférieures et supérieures de même ordre exponentiel sur sa taille minimale:

Théorème 7 *Soit, pour tout $w < r < n$, $b(n,r,w)$ le cardinal minimal d'un $r \Leftrightarrow$recouvrement de la sphère de Hamming de rayon w sur $\{1...n\}$. On a:*

$$\frac{\binom{n}{w}}{\binom{r}{w}} \leq b(n,r,w) \leq \lfloor \ln \binom{r}{w} + 1 \rfloor \frac{\binom{n}{w}}{\binom{r}{w}}$$

$$si\ \Omega < \rho < 1 \qquad b(n, \rho n, \Omega n) \stackrel{exp}{=} 2^{nH_2(\Omega) - \rho n H_2\left(\frac{\Omega}{\rho}\right)}$$

Le traitement de chaque ensemble d'information se faisant en temps polynômial, le coefficient de complexité de l'algorithme déterministe serait donc (avec $r = n \Leftrightarrow k$ et $w = nH_2^{-1}(1 \Leftrightarrow R)$, ou encore $\rho = H_2(\Omega) = 1 \Leftrightarrow R$):

$$(1 \Leftrightarrow R) \left(1 \Leftrightarrow H_2 \left(\frac{H_2^{-1}(1 \Leftrightarrow R)}{1 \Leftrightarrow R}\right)\right)$$

Malheureusement, on ne sait comment générer une liste d'ensembles d'information ayant la propriété requise et une taille du même ordre exponentiel que les bornes mentionnées ci-dessus. Nous allons donc nous intéresser à la version probabiliste.

En supposant toujours l'erreur de poids w, la probabilité pour qu'un ensemble d'information donné soit dépourvu d'erreur vaut $\binom{n-w}{k}/\binom{n}{k} = \binom{n-k}{w}/\binom{n}{w}$. Cette quantité est donc aussi la probabilité p_{succ} pour que le coset leader soit détecté à une itération donnée.

Les itérations étant indépendantes les unes des autres, On en déduit (cf. section 1.3.2, p. 36) que $\frac{n \log n}{p_{succ}}$ itérations suffisent pour effectuer un décodage quasi-complet.

Et puisque $p_{succ} = \binom{n-k}{w}/\binom{n}{w} \stackrel{\exp}{=} 2^{n\left((1-R)H_2\left(\frac{\Omega}{1-R}\right)-H_2(\Omega)\right)}$ (où $\Omega = \frac{w}{n}$), le nombre d'itérations est d'ordre exponentiel:

$$2^{n\left(H_2(\Omega)-(1-R)H_2\left(\frac{\Omega}{1-R}\right)\right)}$$

Le coefficient de complexité est donc (avec $\Omega = H_2^{-1}(1 \Leftrightarrow R)$):

$$(1 \Leftrightarrow R)\left(1 \Leftrightarrow H_2\left(\frac{H_2^{-1}(1 \Leftrightarrow R)}{1 \Leftrightarrow R}\right)\right)$$

c.a.d. le même que dans le cas déterministe.

2.3 Décodage mixte

2.3.1 Recherche de motifs d'erreur: approche directe

Dans l'algorithme décrit en section précédente, un seul élément du coset est calculé à chaque itération. Celui-ci supporte donc seul le coût du pivot. Il est plus intéressant de partager ce coût entre un nombre $T > 1$ d'éléments du coset en autorisant un petit nombre d'erreur p sur l'ensemble d'information.

Par exemple, un élément du coset de poids 1 sur l'ensemble d'information est égal à la somme de l'élément m du coset de poids zéro sur cet ensemble et de l'une des lignes de la matrice génératrice. A tout élément c du coset de poids p sur l'ensemble d'information correspond un ensemble $J \subset \{1...k\}$ de cardinal p tel que $c = m + \sum_{j \in J} g_j$.

Comment évolue la complexité si l'on calcule à chaque itération les $T = \sum_{i=0}^{p} \binom{k}{i} \approx \frac{(nR)^p}{p!}$ éléments du coset de plus petit poids sur l'ensemble d'information ?

La probabilité pour que le coset leader, qui est de poids $w = n\Omega$, soit de poids au plus p sur un ensemble d'information donné, vaut $p_{succ} = \frac{\sum_{i=0}^{p}\binom{n-k}{w-i}\binom{k}{i}}{\binom{n}{w}}$; p_{succ} est donc $\frac{\sum_{i=0}^{p}\binom{n-k}{w-i}\binom{k}{i}}{\binom{n-k}{w}} \approx \frac{\left(\frac{R\Omega}{1-R-\Omega}\right)^p}{p!} n^p$ fois supérieure à la probabilité de succès d'une itération de l'algorithme de base. On peut donc s'attendre à ce que le nombre d'itérations nécessaire à la quasi-complétude soit d'autant diminué.

Chaque itération demande une diagonalisation de la matrice génératrice selon un nouvel ensemble d'information, ce qui coûte en moyenne $n^3R^2(1 \Leftrightarrow R)^2/2$, ainsi que le calcul des T motifs d'erreur (éléments du coset) de poids i ($i \leq p$) sur l'ensemble d'information, et de leur poids, ce qui coûte, pour chacun d'entre eux $(i+1)n(1 \Leftrightarrow R)$.

Le coût de chaque itération est donc multiplié par environ $1+\frac{2T(p+1)n(1-R)}{n^3R^2(1-R)^2} \approx 1 + \frac{2(p+1)(nR)^{p-2}}{(1-R)p!}$.

Le coût global est donc multiplié par:

$$m_p = \frac{p!}{\left(\frac{R\Omega}{1-R-\Omega}\right)^p n^p} \left(1 + \frac{2(p+1)(nR)^{p-2}}{(1 \Leftrightarrow R)p!}\right)$$

On trouve que $m_{p+1} \Leftrightarrow m_p$ est proportionnel à:

$$l_p = \frac{(p+1)!}{(nR)^{p-1}} \left(\frac{1 \Leftrightarrow R}{\Omega} \Leftrightarrow 1\right) \Leftrightarrow \frac{p!}{(nR)^{p-2}} + \frac{2(p+2)}{\Omega} \Leftrightarrow \frac{2(2p+3)}{1 \Leftrightarrow R}$$

La valeur optimale de p est donc le plus petit p tel que $l_p > 0$.

Or $l_0 = nR\left(\frac{1-R}{\Omega} \Leftrightarrow 1\right) \Leftrightarrow (nR)^2 + \frac{4}{\Omega} \Leftrightarrow \frac{6}{1-R}$, donc:

$$nR > \frac{1 \Leftrightarrow R \Leftrightarrow \Omega}{2\Omega} \left(1 + \sqrt{1 + \frac{8\Omega}{(1 \Leftrightarrow R)(1 \Leftrightarrow R \Leftrightarrow \Omega)} \left(2 \Leftrightarrow \frac{\Omega}{1 \Leftrightarrow R \Leftrightarrow \Omega}\right)}\right) \Rightarrow l_0 < 0$$

De même $l_1 = 2\left(\frac{1-R}{\Omega} \Leftrightarrow 1\right) \Leftrightarrow nR + \frac{6}{\Omega} \Leftrightarrow \frac{10}{1-R}$, donc:

$$nR > 2\left(\frac{1 \Leftrightarrow R}{\Omega} \Leftrightarrow 1\right) + \frac{6}{\Omega} \Leftrightarrow \frac{10}{1 \Leftrightarrow R} \Rightarrow l_1 < 0$$

En revanche, si $nR \geq p \geq 2$, alors $\frac{p!}{(nR)^{p-2}} \leq 2$, et puisque $\Omega < \frac{1-R}{2}$, on a:

$$\frac{2(p+2)}{\Omega} \Leftrightarrow \frac{2(2p+3)}{1 \Leftrightarrow R} > \frac{2}{1 \Leftrightarrow R} > 2, \quad \text{et donc } l_p > 0.$$

La valeur optimale de p est donc généralement égale à 2, et ce, indépendamment de n. Le coefficient de complexité ne sera donc pas modifié, mais pour $p = 2$, la complexité est divisée par le polynôme:

$$\frac{1 \Leftrightarrow R}{8 \Leftrightarrow 2R} \left(\frac{R\Omega}{1 \Leftrightarrow R \Leftrightarrow \Omega}\right)^2 n^2$$

On a vu cependant, dans la section traitant des algorithmes de recherche de motifs d'erreur, que la recherche pouvait être accélérée par quelques astuces. La première amélioration, consistant à se restreindre à un ensemble d'information, n'est évidemment pas applicable au décodage par ensemble d'information. La deuxième ne l'est pas directement car on ne connaît pas le syndrôme de l'erreur restreinte à l'ensemble d'information considéré. Mais il est possible d'adapter la troisième.

2.3.2 Utilisation de codes poinçonnés

A chaque itération, plutôt que de calculer les éléments du cosets de poids au plus p sur un ensemble d'information, nous allons chercher ceux qui sont de poids au plus p sur un ensemble contenant strictement un ensemble d'information. Appelons cet ensemble I, $s > k$ son cardinal et supposons s et p pairs, et $p < \frac{ws}{n}$ (si $p = \frac{ws}{n}$, on retrouve le décodage de la section 2.1.4).

Chaque itération consistera en une execution de l'algorithme décrit en section 2.1.4. La différence entre les deux algorithmes réside en ce que le premier est un algorithme déterministe de décodage complet calculant les mots poinçonnés de poids au plus $w\frac{s}{n}$ sur $O(n)$ sous-ensembles déterminés judicieusement; tandis que le second est un algorithme probabiliste de décodage quasi-complet calculant les mots poinçonnés de poids au plus $p < w\frac{s}{n}$ sur un nombre exponentiel de sous-ensembles déterminés aléatoirement.

Ce principe a été proposé par Dumer en 91 [Dum91]. Calculons l'ordre exponentiel du coût global de cet algorithme.

La probabilité de succès à chaque itération est la probabilité pour que le coset leader soit de poids au plus $p/2$ sur deux sous-ensembles du support, aléatoires, disjoints, de cardinal $s/2$:

$$p_{succ} = \frac{\sum_{i=0}^{p/2}\sum_{j=0}^{p/2}\binom{s/2}{i}\binom{s/2}{j}\binom{n-s}{w-i-j}}{\binom{n}{w}} \stackrel{\exp}{=} 2^{sH_2\left(\frac{p}{s}\right)+(n-s)H_2\left(\frac{w-p}{n-s}\right)-nH_2\left(\frac{w}{n}\right)}$$

$\left(\sum_{i=0}^{p/2}\sum_{j=0}^{p/2}\binom{s/2}{i}\binom{s/2}{j}\binom{n-s}{w-i-j} \stackrel{\exp}{=} \max_{0\leq i,j\leq p/2}\binom{s/2}{i}\binom{s/2}{j}\binom{n-s}{w-i-j}\right.$, et on pourrait montrer que ce produit de coefficients binomiaux augmente avec i et j jusqu'à $i = j = \frac{ws}{2n}$, or $\frac{p}{2} < \frac{ws}{2n}$, le max est donc atteint pour $i = j = p/2$, et $\left.\binom{s/2}{p/2}^2\binom{n-s}{w-p} \stackrel{\exp}{=} 2^{sH_2\left(\frac{p}{s}\right)+(n-s)H_2\left(\frac{w-p}{n-s}\right)}\right)$

Le nombre d'itérations nécessaire au décodage quasi-complet est donc d'ordre exponentiel $2^{nH_2\left(\frac{w}{n}\right)-sH_2\left(\frac{p}{s}\right)-(n-s)H_2\left(\frac{w-p}{n-s}\right)}$.

Le coût d'une itération se décompose en:

- Choix d'un s-sous-ensemble contenant un ensemble d'information; pivot sur celui-ci; calcul d'une matrice de parité du code poinçonné. Coût polynomial.

- Calcul et stockage pour chaque moitié de l'ensemble I des $\sum_{i=0}^{p/2}\binom{s/2}{i}$ motifs d'erreur de poids au plus $p/2$ et de leur syndrôme partiel. Coût d'ordre exponentiel $2^{\frac{s}{2}H_2\left(\frac{p}{s}\right)}$.

- Calcul, pour chaque paire de demi-motifs dont les syndrômes partiels se somment en $\tilde{s}$ de l'élément associé du coset et de son poids. Coût d'ordre exponentiel $2^{sH_2\left(\frac{p}{s}\right)-(s-k)}$ (cf. section 2.1.4).

Il est donc d'ordre exponentiel: $2^{\max\left(\frac{s}{2}H_2\left(\frac{p}{s}\right),sH_2\left(\frac{p}{s}\right)-(s-k)\right)}$

Et le coefficient de complexité , $(p/n, s/n)$ de l'algorithme est donc:

$$H_2\left(\frac{w}{n}\right)\Leftrightarrow\frac{s}{n}H_2\left(\frac{p}{s}\right)\Leftrightarrow\frac{n\Leftrightarrow s}{n}H_2\left(\frac{w\Leftrightarrow p}{n\Leftrightarrow s}\right)+\max\left(\frac{s}{2n}H_2\left(\frac{p}{s}\right),\frac{s}{n}H_2\left(\frac{p}{s}\right)\Leftrightarrow\frac{s\Leftrightarrow k}{n}\right)$$

supposons que $w = nH_2^{-1}(1\Leftrightarrow R)$, $s = n\sigma$ et $p = n\pi$, on a alors:

$$\begin{aligned}, (\pi,\sigma) &= (1\Leftrightarrow R)\Leftrightarrow\sigma H_2\left(\frac{\pi}{\sigma}\right)\Leftrightarrow(1\Leftrightarrow\sigma)H_2\left(\frac{H_2^{-1}(1\Leftrightarrow R)\Leftrightarrow\pi}{1\Leftrightarrow\sigma}\right)\\ &\qquad+\max\left(\frac{\sigma}{2}H_2\left(\frac{\pi}{\sigma}\right),\sigma H_2\left(\frac{\pi}{\sigma}\right)\Leftrightarrow\sigma+R\right)\\ &= (1\Leftrightarrow\sigma)\left(1\Leftrightarrow H_2\left(\frac{H_2^{-1}(1\Leftrightarrow R)\Leftrightarrow\pi}{1\Leftrightarrow\sigma}\right)\right)+\max\left(\sigma\Leftrightarrow R\Leftrightarrow\frac{\sigma}{2}H_2\left(\frac{\pi}{\sigma}\right),0\right)\end{aligned}$$

Il s'agit donc de choisir π et σ qui minimisent cette expression et soumis bien-sûr aux contraintes:

$$R\le\sigma<1;\quad 0\le\pi<H_2^{-1}(1\Leftrightarrow R);\quad 0<\sigma\Leftrightarrow\pi<1\Leftrightarrow H_2^{-1}(1\Leftrightarrow R).$$

On remarquera que pour $\sigma = R$ et $\pi = 0$, la complexité est celle du décodage par ensemble d'information.

Au voisinage (à droite) de $\sigma = R$, on a $\max\left(\sigma\Leftrightarrow R\Leftrightarrow\frac{\sigma}{2}H_2\left(\frac{\pi}{\sigma}\right),0\right) = 0$. Lorsque ceci est vérifié, on a:

$$\frac{\partial,}{\partial\sigma}(\pi,\sigma)=\Leftrightarrow 1\Leftrightarrow\log_2\left(1\Leftrightarrow\frac{H_2^{-1}(1\Leftrightarrow R)\Leftrightarrow\pi}{1\Leftrightarrow\sigma}\right)$$

Mais pour $0 < R < 1$, $H_2^{-1}(1\Leftrightarrow R) < \frac{1-R}{2}$; donc au voisinage de $\sigma = R$, $\frac{H_2^{-1}(1-R)-\pi}{1-\sigma} < \frac{1}{2}$ ce qui implique que $\frac{\partial\Gamma}{\partial\sigma}(\pi,R) < 0$, donc que le σ optimal est strictement supérieur à R et surtout que la complexité est exponentiellement inférieure à celle du décodage par ensemble d'information.

La surface représentant , (π,σ) en fonction de π et σ est C^∞ par morceaux, et composée de deux « morceaux » de part et d'autre de la courbe délimitant les régions où le maximum figurant dans l'expression est atteint par l'un ou l'autre de ses deux opérandes.

On pourra résoudre ce problème de minimisation numériquement, et constater que le minimum est atteint précisément sur cette courbe. Supposons donc, afin de pousser l'analyse, et sans chercher à démontrer cette dernière affirmation, que l'on se trouve sur cette frontière, donc que $\sigma\Leftrightarrow R\Leftrightarrow\frac{\sigma}{2}H_2\left(\frac{\pi}{\sigma}\right) = 0$; c.a.d. que $\pi = \sigma H_2^{-1}\left(2\left(1\Leftrightarrow\frac{R}{\sigma}\right)\right)$. Alors $\sum_{i=0}^{p/2}\binom{s/2}{i}\overset{exp}{=}2^{s-k}$ et:

$$, (\sigma) = (1\Leftrightarrow\sigma)\left(1\Leftrightarrow H_2\left(\frac{H_2^{-1}(1\Leftrightarrow R)\Leftrightarrow\sigma H_2^{-1}\left(2\left(1\Leftrightarrow\frac{R}{\sigma}\right)\right)}{1\Leftrightarrow\sigma}\right)\right)$$

σ est alors soumis aux contraintes:

$$R<\sigma<\min(1,2R);\quad \sigma H_2^{-1}\left(2\left(1\Leftrightarrow\frac{R}{\sigma}\right)\right)<H_2^{-1}(1\Leftrightarrow R);$$

$$\sigma\left(1 \Leftrightarrow H_2^{-1}\left(2\left(1 \Leftrightarrow \frac{R}{\sigma}\right)\right)\right) < 1 \Leftrightarrow H_2^{-1}(1 \Leftrightarrow R).$$

On essaiera de trouver numériquement l'argument minimum de , , qui déterminera σ et l'on pourra en déduire π.

2.3.3 Utilisation de surcodes

L'optimisation de la complexité asymptotique ne correspond pas toujours aux besoins réels (longueurs finies). Elle conduit parfois à mettre au point des algorithmes qui ont certes un coefficient de complexité intéressant, mais qui sont d'une sophistication telle que l'on se demande à partir de quelle longueur ils commencent à s'avérer plus intéressants que leurs concurrents.

Cela dit, pour être complète, cette section se doit de mentionner l'algorithme trouvé par Barg, Krouk et Van Tilborg en 1999 [BKvT99], dont le coefficient de complexité semble inférieur à celui que nous venons d'étudier et serait donc le plus petit connu à ce jour, et qui est peut-être un exemple du phénomène mentionné ci-dessus.

Il ressemble au précédent en ce qu'il se repose sur le décodage par ensemble d'information, en consacrant un effort exponentiel à chaque itération, de manière à pouvoir « capturer » le plus grand nombre possible de motifs d'erreur en un coût minimal, et en utilisant pour ce faire des techniques de syndrômes partiels.

Son originalité réside dans le calcul d'intersections d'ensembles construits dans une première phase de l'algorithme. Ces intersections détermineront un ensemble de candidats.

Soient w le poids de l'erreur, $0 \leq p \leq w$, $0 < y < n \Leftrightarrow k$ et $1 \leq b \leq 1 + \frac{n-k-w+p}{y}$ des paramètres de l'algorithme. Notons tout de suite que p/n, y/n et b seront optimisés à la fin de cette étude indépendamment de n, et ne dépendront que du taux de codage R.

A chaque itération, nous allons considérer une partition du support en l'ensemble d'information d'une part et en $s = \frac{n-k}{y}$ sous-ensembles de cardinal y d'autre part (on supposera que y divise $n \Leftrightarrow k$); et nous allons calculer l'ensemble E des éléments du coset de poids p (et non pas « au plus p ») sur l'ensemble d'information et de poids au plus $q = \frac{w-p}{s-b+1}$ sur au moins b des s sous-ensembles de taille y.

En effet, si l'erreur, de poids w, est de poids p sur l'ensemble d'information, alors elle est de poids au plus q sur au moins b des s sous-ensembles de taille y. Sinon elle serait de poids supérieur à q sur au moins $s \Leftrightarrow b + 1$ de ces ensembles et par définition de q elle serait de poids total supérieur à w. La probabilité pour que l'erreur appartienne à l'ensemble E est donc la probabilité pour qu'elle soit de poids p sur l'ensemble d'information. On a donc:

$$p_{succ} = \frac{\binom{k}{p}\binom{n-k}{w-p}}{\binom{n}{w}} \stackrel{\exp}{=} 2^{-c_1\left(R, \frac{w}{n}, \frac{p}{n}\right)n} \tag{2.1}$$

avec $c_1(R, \Omega, \pi) = H_2(\Omega) \Leftrightarrow R H_2\left(\frac{\pi}{R}\right) \Leftrightarrow (1 \Leftrightarrow R) H_2\left(\frac{\Omega - \pi}{1 - R}\right)$.

Il s'agit donc de calculer l'ensemble E à un coût minimal. Nous allons utiliser des techniques de séparation de syndrômes.

Supposons, sans perte de généralité, que l'ensemble d'information soit formé des k positions les plus à gauche, que les s sous-ensembles aillent à sa suite de manière connexe, et appelons S_i $(i = 1...s)$ l'ensemble réunion de l'ensemble d'information et du $i$ème sous-ensemble. Soient $H = [A|I_{n-k}]$ une matrice de parité du code sous forme systématique à droite, et pour $i = 1...s$, $H_i = [A_i|I_y]$ la $y \times (k+y)$ matrice composée des lignes de H correspondant au $i$ème sous-ensemble et de support égal à S_i; et pour tout mot x, soit $s_i(x)$ le syndrôme selon cette matrice de x restreint ou étendu au support S_i.

Tout d'abord, considérons la partition de l'ensemble d'information ($\{1...k/2\}$, $\{k/2+1...k\}$) et tous ses « shifts » modulo k, soit $k/2$ partitions en tout. Tout motif de poids p sur l'ensemble d'information se sépare en deux motifs de poids $p/2$ sur au moins l'une d'entre elles. Nous allons donc effectuer $k/2$ fois le même travail, ce qui ne changera pas l'ordre exponentiel de la complexité. Nous ne décrirons ce travail que pour la première partition.

Pour $i = 1...s$, nous allons calculer et stocker l'ensemble de tous les motifs de poids $p/2$ sur la partie gauche ainsi que leur syndrôme selon H_i, puis nous ferons de même pour la partie droite. On obtient ainsi pour chaque i deux tables $I_g^{(i)}$ et $I_d^{(i)}$:

$$I_g^{(i)} = \{(s_{Ig}, e_{Ig}) / \operatorname{wt}(e_{Ig}) = p/2, s_i(e_{Ig}) = s_{Ig}\}$$

$$I_d^{(i)} = \{(s_{Id}, e_{Id}) / \operatorname{wt}(e_{Id}) = p/2, s_i(e_{Id}) = s_{Id}\}$$

La complexité en temps et en mémoire de cette construction est d'ordre exponentiel:

$$2^{\frac{k}{2}H_2(\frac{p}{k})}$$

En séparant de même chaque y-sous-ensemble en deux (de $y/2$ manières différentes), on calcule et on stocke, pour $i = 1...s$ et pour chaque moitié de sous-ensemble, des motifs de poids au plus $q/2$ et leur syndrômes selon H_i (cette partie de H_i étant diagonale, les motifs sont égaux à leur syndrôme). On obtient ainsi deux tables $R_g^{(i)}$ et $R_d^{(i)}$:

$$R_g^{(i)} = \{(s_{Rg}, e_{Rg}) / \operatorname{wt}(e_{Rg}) \leq q/2, s_i(e_{Rg}) = s_{Rg}\}$$

$$R_d^{(i)} = \{(s_{Rd}, e_{Rd}) / \operatorname{wt}(e_{Rd}) \leq q/2, s_i(e_{Rd}) = s_{Rd}\}$$

La complexité en temps et en mémoire de cette construction est d'ordre exponentiel:

$$2^{\frac{y}{2}H_2(\frac{q}{y})}$$

Enfin nous construisons les tables $IR_g^{(i)}$ et $IR_d^{(i)}$ définis par:

$$IR_g^{(i)} = \{(s_g, e_{Ig}) / \exists s_{Ig}, s_{Rg}, e_{Rg} \quad s_{Ig} + s_{Rg} = s_g, (s_{Ig}, e_{Ig}) \in I_g^{(i)}, (s_{Rg}, e_{Rg}) \in R_g^{(i)}\}$$

$$IR_d^{(i)} = \{(s_d, e_{Id}) / \exists s_{Id}, s_{Rd}, e_{Rd} \quad s_{Id} + s_{Rd} = s_d, (s_{Id}, e_{Id}) \in I_d^{(i)}, (s_{Rd}, e_{Rd}) \in R_d^{(i)}\}$$

Et nous trions ces tables par ordre croissant de s (cette construction est à faire pour chaque choix de partitions, soit de $\frac{k}{2}\frac{y}{2}$ manières différentes).

La complexité en temps et en mémoire de cette construction est d'ordre exponentiel:

$$2^{\frac{k}{2}H_2\left(\frac{p}{k}\right)+\frac{y}{2}H_2\left(\frac{q}{y}\right)} = 2^{\frac{1}{2}c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)n} \tag{2.2}$$

avec $c_2(R,\Omega,\pi,\nu,b) = RH_2\left(\frac{\pi}{R}\right) + \nu H_2\left(\frac{\Omega-\pi}{1-R-(b-1)\nu}\right)$.

Pour tout syndrôme partiel $s_g \in \mathbf{F}_2^y$ apparaissant dans $IR_g^{(i)}$, on définit les tables:

$$T_g^{(i)}(s_g) = \{e_{Ig}/(s_g, e_{Ig}) \in IR_g^{(i)}\}$$

$$T_d^{(i)}(s_g) = \{e_{Id}/(s_g + s_i(m), e_{Id}) \in IR_d^{(i)}\}$$

La phase suivante est à executer pour chacun des $\binom{s}{b}$ groupes $1 \leq i_1 < i_2 < ... < i_b \leq s$ de b sous-ensembles (s et b ne dépendent pas de n, $\binom{s}{b}$ est uniquement fonction de R et n'a pas d'influence sur l'ordre exponentiel des complexités). Supposons, sans perte de généralité, que $i_1 = 1, i_2 = 2, ..., i_b = b$.

Appelons K_i la table suivante que nous n'essaierons pas de construire:

$$K_i = \{(e_{Ig}, e_{Id})/\exists s_g \quad e_{Ig} \in T_g^{(i)}(s_g), e_{Id} \in T_d^{(i)}(s_g)\}$$

K_i peut se déduire de $IR_g^{(i)}$ et $IR_d^{(i)}$ qui sont de taille exponentiellement plus petite. Nous dirons que l'on en a une forme séparée.

Il s'agit de calculer l'ensemble $\cap_{i=1}^b K_i$, ce qui nous permettra de construire l'ensemble E recherché (réunion des $\binom{s}{b}$ telles intersections), mais nous allons faire en sorte de ne travailler qu'avec des formes séparées.

Pour construire cette intersection, nous procédons en $b \Leftrightarrow 1$ étapes analogues : On construit une forme séparée de $K_{12} = K_1 \cap K_2$, de même pour $K_{123} = K_{12} \cap K_3$, et ainsi de suite jusqu'à $K_{12..b}$ qui est bien-sûr égal à $\cap_{i=1}^b K_i$. Il nous suffit donc de décrire la construction d'une forme séparée pour K_{12}.

Pour tout couple $(s_g^{(1)}, s_g^{(2)})$, on construit les tables:

$$T_g^{(12)}(s_g^{(1)}, s_g^{(2)}) = T_g^{(1)}(s_g^{(1)}) \cap T_g^{(2)}(s_g^{(2)})$$

$$T_d^{(12)}(s_g^{(1)}, s_g^{(2)}) = T_d^{(1)}(s_g^{(1)}) \cap T_d^{(2)}(s_g^{(2)})$$

Les couples $(s_g^{(1)}, s_g^{(2)})$ tels que l'une au moins de ces deux tables est vide sont éliminés.

Le coût de cette construction est d'ordre exponentiel:

$$2^{\frac{1}{2}c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)n+y} \tag{2.3}$$

$K_1 \cap K_2$ est alors déterminé par:

$$K_{12} = \{(e_{Ig}, e_{Id})/\exists s_g^{(1)}, s_g^{(2)} \quad e_{Ig} \in T_g^{(12)}(s_g^{(1)}, s_g^{(2)}), e_{Id} \in T_d^{(12)}(s_g^{(1)}, s_g^{(2)})\}$$

On construit ensuite de la même manière et pour un coût analogue ou inférieur (la somme des tailles des tables T_g et T_d diminuant au fur et à mesure), des formes séparés de $K_{123}...K_{12...b}$. Enfin on construit $K_{12...b}$.

La taille de cette dernière table, égale au coût de sa construction, est d'ordre exponentiel:

$$2^{c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)n-by} \tag{2.4}$$

Finalement, au regard de 2.1, 2.2, 2.3 et 2.4 la complexité globale est d'ordre exponentiel:

$$2^{n\left(c_1\left(R,\frac{w}{n},\frac{p}{n}\right)+\max\left(\frac{1}{2}c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)+\frac{y}{n},c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)-b\frac{y}{n}\right)\right)}$$

Et $c_1\left(R,\frac{w}{n},\frac{p}{n}\right)+\max\left(\frac{1}{2}c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)+\frac{y}{n},c_2\left(R,\frac{w}{n},\frac{p}{n},\frac{y}{n},b\right)\Leftrightarrow b\frac{y}{n}\right)$ est le coefficient de complexité de l'algorithme. On choisira bien-sûr p, y et b de manière à le minimiser dans le cas $\frac{w}{n}=H_2^{-1}(1\Leftrightarrow R)$. Le résultat de cette minimisation est le plus petit coefficient de complexité que l'on connaisse pour le décodage quasi-complet (cf. figure 2.1).

Cependant, le terme constant et le polynôme figurant dans la complexité réelle sont si gros que cet algorithme ne devient plus intéressant que son concurrent de la section précédente que pour des longueurs de toute façon trop importante pour que l'on puisse les décoder.

2.4 Nombre d'itérations des algorithmes n'effectuant que des pivots vicinaux

Lorsque l'algorithme considère un nouvel ensemble d'information, il doit effectuer une diagonalisation (un pivot) de la matrice génératrice selon cet ensemble d'information, ce qui a un coût de $n^3\frac{R^2(1-R)^2}{2}$.

Mais plutôt que de considérer un ensemble d'information totalement indépendant du précédent, il est possible d'en choisir un qui ne diffère du précédent que par une position. On parle alors d'ensembles d'information « vicinaux » et de pivot « vicinal ».

L'intérêt est que le pivot vicinal a un coût $n^2\frac{R(1-R)}{2}$ en moyenne, soit $nR(1\Leftrightarrow R)$ fois plus faible que le coût moyen du pivot général. Lorsque le coût relatif des pivots n'est pas négligeable dans l'algorithme, il peut donc être valable d'explorer successivement des ensembles d'information vicinaux.

Cette variante avait déjà été proposé par Lee et Brickell [LB88] comme une amélioration possible de leur algorithme et a été mis en oeuvre en 1994 par Anne Canteaut, Florent Chabaud et Hervé Chabanne [CC94] [CC95] qui l'ont combinée avec une méthode proposée par Stern en 89 [Ste89] ressemblant à l'utilisation de codes poinçonnés [1].

1. Il est intéressant de remarquer qu'une fois de plus, c'est dans le but de « casser » des cryptosystèmes basés sur les codes et non dans un but lié à la correction d'erreur de transmission, que ces nouveaux algorithmes ont été proposés.

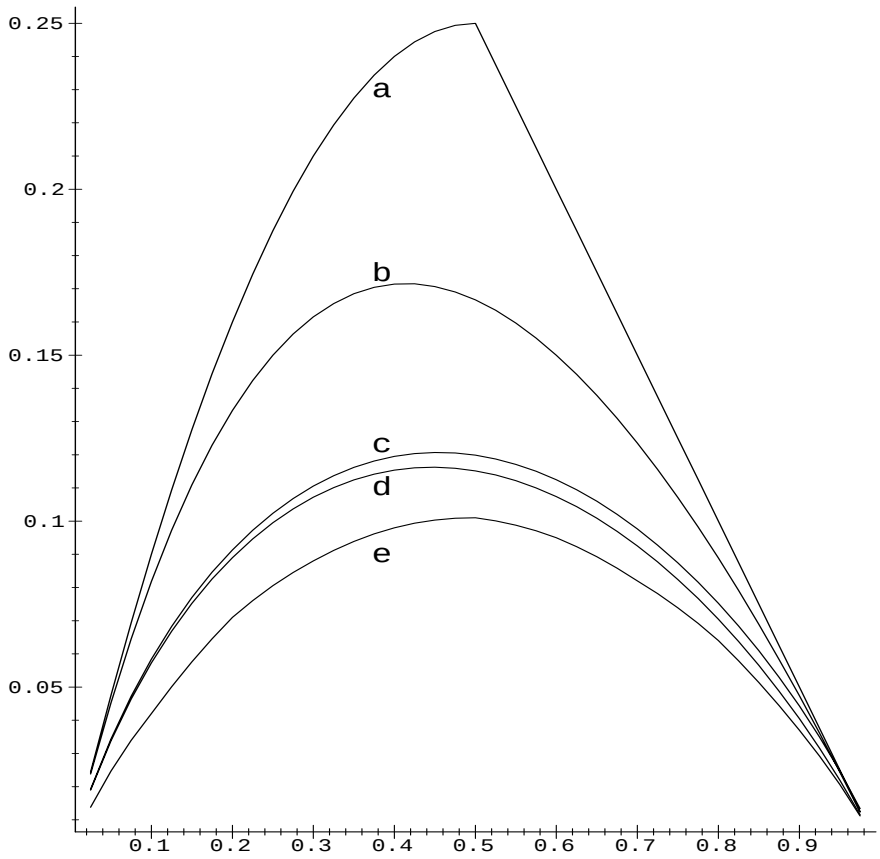


FIG. 2.1: *Coefficient de complexité des différents algorithmes: (a)* $\min(R(1 \Leftrightarrow R), (1 \Leftrightarrow R)/2)$*; (b)* $R(1 \Leftrightarrow R)/(1+R)$*; (c) décodage par ensemble d'information; (d) idem avec utilisation de codes poinçonnés; (e) idem avec utilisation de surcodes*

Le fait de ne permuter que deux positions entre chaque itération a pour conséquence de rendre les itérations successives dépendantes. Et l'étude faite en section 1.3.2, p. 36, n'est donc plus valable.

Entre chaque itération, donc, on effectue un pivot vicinal. Un choix doit être fait quant aux deux positions à permuter. Il y a autant de choix possibles que de « 1's » dans la partie redondante de la matrice génératrice, soit environ $\frac{k(n-k)}{2}$.

Contrairement à ce qu'il se passe pour le décodage souple, on n'a, dans le cas du décodage dur, aucune information a priori sur la localisation des erreurs de transmission. Pour être plus précis, la seule information dont on puisse disposer est la parité du nombre d'erreurs contenues dans le support des différents mots du dual. Mais cette information n'apporte quasiment rien dès lors que l'espérance du nombre d'erreurs sur le plus petit support est supérieure à 1 (d'où l'intérêt des LDPC codes de Gallager[1] pour lesquels certains supports de mots du dual peuvent contenir moins de 1 erreur en moyenne). Les différentes possibilités pour le choix des positions à permuter paraissent donc a priori équivalentes.

Deux alternatives pour faire ces choix: ou bien on parcourt les différents ensembles d'information d'une manière déterministe; ou bien on permute à chaque itération deux positions choisies aléatoirement.

Une méthode déterministe requiert ou bien une liste préétablie de permutations à effectuer, ou bien une heuristique permettant de décider quelles positions permuter. A ce jour, ces possibilités n'ont pas encore donné lieu à des résultats probants. J'ai personnellement réussi à diminuer le nombre d'itérations moyen, mais le gain relatif obtenu en terme de nombre d'itérations est toujours resté inférieur au coût relatif des routines d'aide à la décision, augmentant donc le coût global.

Nous allons donc étudier le nombre d'itérations et la probabilité de succès des algorithmes de décodage à base d'ensemble d'information dans le cas où les permutations sont aléatoires. Nous confondrons motif d'erreur et coset leader, c.a.d. que nous dirons qu'une position est erronée ou contient une erreur si le coset leader vaut 1 en cette position. Par ailleurs nous supposerons que w, le poids du coset leader est inférieur à $\min(k, n \Leftrightarrow k)$ (on peut avoir $w > k$ pour des taux de codage inférieurs à 23%).

L'implémentation basique garantit le succès d'une itération si et seulement si l'ensemble d'information est complètement dépourvu d'erreur. D'autres implémentations sont telles que le coset leader sera calculé si l'ensemble d'informations contient au plus q erreurs. Dans le cas général, si lors d'une itération donnée, l'ensemble d'information contient u erreur, alors le coset leader sera calculé avec une probabilité p_u. Quasiment tous les algorithmes de décodage à base d'ensemble d'information peuvent se ramener à ce cas, il suffit de donner les valeurs adéquates à $(p_u)_{0 \leq u \leq w}$.

Soient donc, en fonction de l'implémentation choisie, $p_u \in [0,1]; u = 0...w$ les probabilités pour que le coset leader soit calculé au cours de l'itération si l'ensemble d'information contient u erreurs.

Nous allons essayer de donner (en fonction de n, k, w, q, (p_u) et N) la

1. Low Density Parity Check Codes

probabilité pour que le coset leader ait été calculé au bout de N itérations.

2.4.1 modélisation par un processus markovien

On associe à chaque itération i de l'algorithme une variable aléatoire $U_i \in \{0...w\}$ représentant le nombre d'erreurs dans l'ensemble d'infomation. La séquence $(U_i)_{i=0,1...}$ est ce que l'on appelle une chaîne de Markov.

Soit P la $((w+1)\times(w+1))$-matrice, dite de transition, vérifiant:

$$\forall 0 \leq u, v \leq w \quad P_{u,v} = \Pr(U_i = v | U_{i-1} = u)$$

Remarque 2 *De part le fait que la somme des probabilités d'évènements complémentaires vaut nécessairement 1, la somme des coefficients de chaque ligne de la matrice P est égale à 1. Si l'on note $\bar{1}$ le vecteur colonne de taille $w+1$, composés uniquement de 1, on a donc $P\bar{1} = \bar{1}$. c.a.d. que P admet une valeur propre égale à 1 (on peut également montrer que le rayon spectral de P est égal à 1) et que $\bar{1}$ est un vecteur propre droite de P associé à cette valeur propre.*

L'intérêt de cette matrice est que si l'on dispose de la distribution de probabilité $\pi^{i-1} = (\Pr(U_{i-1} = u))_{0\leq u\leq w}$ mise sous forme de vecteur ligne, on en déduit la distribution $\pi^i = (\Pr(U_i = u))_{0\leq u\leq w}$ par:

$$\pi^i = \pi^{i-1} P$$

Puisqu'entre chaque itération on ne permute que deux positions, la variable aléatoire U_i changera d'au plus une unité. La matrice P est donc tridiagonale:

$$\forall u \in \{0...w\} \quad \Pr(U_i > u+1 | U_{i-1} = u) = \Pr(U_i < u \Leftrightarrow 1 | U_{i-1} = u) = 0$$

La permutation consiste à échanger deux positions, celle qui passe de l'ensemble d'information à son complémentaire sera appelée position rejetée, l'autre sera appelée position admise.

Si l'ensemble d'information contient u erreurs (et donc son complémentaire $w \Leftrightarrow u$), la probabilité pour que la position rejetée soit une erreur vaut u/k et la probabilité pour que la position admise en soit une vaut $\frac{w-u}{n-k}$. On a donc, pour $u = 0...w$:

$$\begin{aligned}
\beta_{u-1} = P_{u,u-1} = &\quad \Pr(U_i = u \Leftrightarrow 1 | U_{i-1} = u) &&= \frac{u}{k}\left(1 \Leftrightarrow \frac{w \Leftrightarrow u}{n \Leftrightarrow k}\right) \\
\alpha_u = P_{u,u} = &\quad \Pr(U_i = u | U_{i-1} = u) &&= \frac{u}{k}\frac{w \Leftrightarrow u}{n \Leftrightarrow k} + \left(1 \Leftrightarrow \frac{u}{k}\right)\left(1 \Leftrightarrow \frac{w \Leftrightarrow u}{n \Leftrightarrow k}\right) \\
\gamma_u = P_{u,u+1} = &\quad \Pr(U_i = u+1 | U_{i-1} = u) &&= \left(1 \Leftrightarrow \frac{u}{k}\right)\frac{w \Leftrightarrow u}{n \Leftrightarrow k}
\end{aligned}$$

(Les termes, $\beta_{-1} = P_{0,-1}$ et $\gamma_w = P_{w,w+1}$ que cette définition n'exclue pas sont égaux à 0, la matrice P est donc définie sans ambigüité)

On a noté $\alpha_u = P_{u,u}$, $\beta_u = P_{u+1,u}$ et $\gamma_u = P_{u,u+1}$, en sorte que:

$$P = \begin{bmatrix} \alpha_0 & \gamma_0 & & (0) \\ \beta_0 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{w-1} \\ (0) & & \beta_{w-1} & \alpha_w \end{bmatrix}$$

Soit $\pi = \pi^0 = (\pi_u)_{0 \leq u \leq w}$, la distribution initiale des probabilités pour que U soit égal à u. On trouve facilement que

$$\forall 0 \leq u \leq w \quad \pi_u = \frac{\binom{n-w}{k-u}\binom{w}{u}}{\binom{n}{k}} = \frac{\binom{n-k}{w-u}\binom{k}{u}}{\binom{n}{w}}$$

La somme des coefficient du vecteur π est bien-sûr égale à 1.

Remarque 3 *Les permutations étant choisies de manière aléatoire, le fait de changer d'ensemble d'information ne change pas cette distribution, on dit que l'on a stationnarité du processus; en particulier on a donc également une symétrie du processus par inversion temporelle (ce qui implique par exemple que $\beta_u \pi_{u+1} = \gamma_u \pi_u$ ($u = 0...w \Leftrightarrow 1$), ce que le lecteur pourra vérifier).*

Une conséquence de la stationnarité est que $\pi P = \pi$, c.a.d. que π est un vecteur propre gauche de P associé à la valeur propre 1; ce qui est formellement équivalent aux trois propriétés suivantes, aisément vérifiables:

$$\alpha_0 \pi_0 + \beta_0 \pi_1 = \pi_0$$
$$\gamma_{u-1} \pi_{u-1} + \alpha_u \pi_u + \beta_u \pi_{u+1} = \pi_u; u = 1...w \Leftrightarrow 1$$
$$\gamma_{w-1} \pi_{w-1} + \alpha_w \pi_w = \pi_w$$

Les états $U_i = 0...w$ peuvent correspondre soit à un succès (avec une probabilité p_{U_i}), ce que nous noterons $S(U_i)$ soit à un échec (avec une probabilité $1 \Leftrightarrow p_{U_i}$), ce que noterons $E(U_i)$. Nous allons redéfinir la matrice P et le vecteur π de manière à les restreindre aux états d'échec. c.a.d. que nous définissons la $((w+1) \times (w+1))$-matrice Q vérifiant:

$$\forall 0 \leq u, v \leq w \quad Q_{u,v} = \Pr(U_i = v, E(U_i) | U_{i-1} = u)$$

Les colonnes v de Q sont égales à celles de P multipliée par $1 \Leftrightarrow p_v$. Soit, donc la $((w+1) \times (w+1))$-matrice diagonale $E = \mathrm{Diag}(1 \Leftrightarrow p_0, ..., 1 \Leftrightarrow p_w)$, on a: $Q = PE$ (Q est également tridiagonale).

De même, πE représente la distribution de probabilité initiale des états d'échec (on remarque que $\pi E = \pi P E = \pi Q$).

Proposition 7 *Pour tout N, la probabilité pour qu'aucune des N premières itérations n'ait permis de calculer le coset leader vaut:*

$$\Pr(\forall i \leq N \quad E(U_i)) = \pi Q^{N+1} \bar{1}$$

Démonstration: Pour tout $i \leq N$ et pour tout $0 \leq v \leq w$, on définit le vecteur $\mu_i^{(v)}$ par:

$$\mu_i^{(v)} = (\Pr(U_i = u, \forall j \leq i \quad E(U_j)|U_0 = v))_{u=0\ldots w}$$

La probabilité de l'évènement $E(U_i)$ ne dépend par hypothèse que de la valeur de U_i; la distribution de probabilité quant à la valeur de U_i ne dépend quant à elle que de la valeur de U_{i-1}. On a donc, pour tout $0 \leq u \leq w$:

$$\begin{aligned}&\Pr(U_i = u, \forall j \leq i \quad E(U_j)|U_0 = v)\\ &= \sum_{u'=0}^{w} \Pr(U_i = u, E(U_i)|U_{i-1} = u') \Pr(U_{i-1} = u', \forall j \leq i \Leftrightarrow 1 \quad E(U_j))|U_0 = v,)\end{aligned}$$

On en déduit que $\mu_i^{(v)} = \mu_{i-1}^{(v)} Q$.

Mais $\mu_0^{(v)}$ vaut par définition $1 \Leftrightarrow p_v$ en position v et 0 ailleurs, c.a.d. $e_v E$, avec e_v le v-ième vecteur de la base canonique, on déduit que:

$$\mu_i^{(v)} = e_v E Q^i$$

Comme par ailleurs:

$$\begin{aligned}\Pr(U_i = u, \forall j \leq i \quad E(U_j)) &= \sum_{v=0}^{w} \pi_v (\mu_i^{(v)})_u \\ &= \sum_{v=0}^{w} (\pi_v e_v E Q^i)_u = (\pi E Q^i)_u = (\pi Q^{i+1})_u\end{aligned}$$

On a:

$$\Pr(\forall j \leq i \quad E(U_j)) = \sum_{u=0}^{w} (\pi Q^{i+1})_u = \pi Q^{i+1} \bar{1}$$

□

Contrairement à P et π, la somme des coefficients d'une ligne de Q peut être strictement inférieure à 1, en conséquence, et puisque ces sommes ne peuvent être strictement supérieure à 1 en valeur absolue, le rayon spectral de Q est strictement inférieur à 1, et Q^N tend vers la matrice nulle, ceci implique que la probabilité de succès de l'algorithme tend vers 1. On peut donc supposer que l'algorithme, s'il tournait indéfiniment, finirait par calculer le coset leader, ce qui nous permet de définir $\bar{N} \in \mathbf{N}$, la variable aléatoire représentant le numéro de la première itération pendant laquelle le coset leader sera calculé (en supposant toujours que l'algorithme n'est pas bridé et tourne indéfiniment). Et l'on a bien-sûr:

$$\Pr(\forall i \leq N \quad E(U_i)) = \Pr(\bar{N} > N)$$

Pour que le décodage soit quasi-complet, il s'agit alors de dimensionner N, le nombre maximal d'itérations, de manière à ce que $\Pr(\bar{N} > N)$ (qui est donc égal à $\pi Q^{N+1}\bar{1}$) soit du même ordre que P_{ML}, la probabilité pour que le motif d'erreur soit différent du coset leader.

Si ρ est la plus grande valeur propre de Q (que l'algorithme calculera), on pourra prendre $N = \frac{\log P_{ML}}{\log \rho}$ (on pourra approximer P_{ML} par la borne sphérique, cf. section 1.1.4, p. 30), ou encore $N = \frac{n \log n}{1-\rho}$.

Mais voyons ce que valent les deux premiers moments (espérance et variance) de $\bar{N}$:

Proposition 8 *Si l'on note $R = (I \Leftrightarrow Q)^{-1}$, on a:*

$$\mathrm{Esp}(\bar{N}) = \pi R \bar{1} \Leftrightarrow 1$$

$$\mathrm{Var}(\bar{N}) = 2\pi RPR\bar{1} \Leftrightarrow \pi R\bar{1} \Leftrightarrow (\pi R\bar{1})^2$$

Démonstration: L'espérance se calcule facilement:

$$\begin{aligned}
\mathrm{Esp}(\bar{N}) &= \sum_{N=1}^{\infty} N \Pr(\bar{N} = N) = \sum_{N=0}^{\infty} \Pr(\bar{N} > N) \\
&= \sum_{N=0}^{\infty} \pi Q^{N+1}\bar{1} = \pi \left(\sum_{N=0}^{\infty} Q^{N+1} \right) \bar{1} \\
&= \pi \left((I \Leftrightarrow Q)^{-1} \Leftrightarrow I \right) \bar{1} = \pi R\bar{1} \Leftrightarrow 1
\end{aligned}$$

Quant à la variance, on pourra la déduire de l'espérance de $\bar{N}^2$, mais les choses se compliquent.

Appelons M_u^1 et M_u^2 l'espérance, si $U_0 = u$, de $\bar{N}$ et $\bar{N}^2$ respectivement, M^1 le vecteur colonne $(M_u^1)_{u=0\ldots w}$ et M^2 le vecteur colonne, $(M_u^2)_{u=0\ldots w}$. En particulier, en vertu de ce qui précède, $M^1 = \sum_{N=0}^{\infty} Q^{N+1}\bar{1} = (R \Leftrightarrow I)\bar{1}$; par ailleurs, on a bien-sûr $\mathrm{Esp}(\bar{N}) = \pi M^1$ et $\mathrm{Esp}(\bar{N}^2) = \pi M^2$.

Or pour tout u:

$$M_u^2 = \sum_{N=0}^{\infty} N^2 \Pr(\bar{N} = N | U_0 = u) = \sum_{N=0}^{\infty} (N+1)^2 \Pr(\bar{N} = N+1 | U_0 = u)$$

Mais:

$$\begin{aligned}
&\Pr(\bar{N} = N+1 | U_0 = u) \\
&= \Pr(E(U_0) | U_0 = u) \sum_{v=0}^{w} \Pr(U_1 = v | U_0 = u) \Pr(\bar{N} = N+1 | U_1 = v) \\
&= (1 \Leftrightarrow p_u) \sum_{v=0}^{w} P_{u,v} \Pr(\bar{N} = N | U_0 = v)
\end{aligned}$$

Donc:

$$M_u^2 = (1 \Leftrightarrow p_u) \sum_{v=0}^{w} P_{u,v} \sum_{N=0}^{\infty} (N+1)^2 \Pr(\bar{N} = N | U_0 = v)$$
$$= (1 \Leftrightarrow p_u) \sum_{v=0}^{w} P_{u,v} (M_v^2 + 2M_v^1 + 1)$$

On en déduit: $M^2 = EP(M^2 + 2M^1 + \bar{1})$ $(EP \neq PE = Q)$ et donc $M^2 = (I \Leftrightarrow EP)^{-1} EP(2M^1 + \bar{1})$.

Mais $(I \Leftrightarrow EP)^{-1} EP = \sum_{i=0}^{\infty} (EP)^{i+1} = E \sum_{i=0}^{\infty} (PE)^i P = ERP$.

Donc $M^2 = ERP(2M^1 + \bar{1}) = ERP(2R \Leftrightarrow I)\bar{1}$, et:

$$\mathrm{Esp}(\bar{N}^2) = \pi ERP(2R \Leftrightarrow I)\bar{1}$$

Or $\pi E = \pi Q$ et $QR = R \Leftrightarrow I$, donc $\mathrm{Esp}(\bar{N}^2) = \pi(R \Leftrightarrow I)P(2R \Leftrightarrow I)\bar{1}$.

Par ailleurs $\pi P = \pi$, $P\bar{1} = \bar{1}$ et $\pi\bar{1} = 1$ donc $\mathrm{Esp}(\bar{N}^2) = 2\pi RPR\bar{1} \Leftrightarrow 3\pi R\bar{1} + 1$.

Comme $\mathrm{Esp}(\bar{N}) = \pi R\bar{1} \Leftrightarrow 1$, on en déduit que:

$$\mathrm{Var}(\bar{N}) = \mathrm{Esp}(\bar{N}^2) \Leftrightarrow \mathrm{Esp}(\bar{N})^2 = 2\pi RPR\bar{1} \Leftrightarrow (\pi R\bar{1})^2 \Leftrightarrow \pi R\bar{1}$$

□

2.4.2 Résultats explicites dans un cas particulier

Le but de cette section est d'obtenir un élément de comparaison entre le nombre d'itérations des algorithmes effectuant des pivots vicinaux, et celui des mêmes algorithmes effectuant des pivots indépendants.

En fait, nous allons comparer $E(\bar{N})$ le nombre moyen d'itérations avant que le coset leader soit effectivement calculé, qui est différent du nombre d'itérations nécessaire à la quasi-complétude du décodage, mais est tout de même une bonne mesure de la complexité du décodage.

Dans certains cas le nombre moyen d'itérations peut être calculé explicitement. Il s'agit du cas où une condition nécessaire et suffisante de succès d'une itération est que le nombre d'erreur dans l'ensemble d'information soit au plus égal à un certain nombre p donné (dans le cas général, on n'a là qu'une condition nécessaire).

Remarquons d'ors et déjà que dans ce cas, la probabilité de succès lors d'une itération donnée vaut $\pi_0 + ... + \pi_p \approx \pi_p$. On montre facilement que le nombre moyen d'itérations de l'algorithme qui effectuerait des pivots indépendants est l'inverse de cette probabilité.

Pour tout $q = 0...w$, soit E_q la $(w+1) \times (w+1) \Leftrightarrow$matrice diagonale dont les $q+1$ premiers coefficients (sur la diagonale) sont nuls et dont les autres sont égaux à 1. La matrice E définie précédemment $(Q = PE)$ est par définition de la condition de succès égale à E_p.

P étant tridiagonale et puisque $\pi P = \pi$, on a:

$$m < p \Rightarrow \begin{cases} E_m Q = E_m P E_p = P E_p, \pi E_m Q = \pi E_p, \\ \pi E_m (I \Leftrightarrow Q) = \pi (E_m \Leftrightarrow E_p) = [0...0, \pi_{m+1}, ..., \pi_p, 0...0] \end{cases}$$

$$\pi E_p (I \Leftrightarrow Q) = \pi E_p (I \Leftrightarrow P) E_p$$

$$m > p \Rightarrow E_m Q = E_m P E_p = E_m P, \pi E_m (I \Leftrightarrow Q) = \pi E_m (I \Leftrightarrow P)$$

Par ailleurs, sachant que $\gamma_{u-1}\pi_{u-1} + \alpha_u \pi_u + \beta_u \pi_{u+1} = \pi_u ; u = 1...w \Leftrightarrow 1$, on vérifie que:

$$0 < m < w \Rightarrow \pi E_m P = [0...0, \beta_m \pi_{m+1}, \alpha_{m+1}\pi_{m+1} + \beta_{m+1}\pi_{m+2}, \pi_{m+2}, ..., \pi_w]$$

Et puisque $\beta_m \pi_{m+1} = \gamma_m \pi_m$, on a:

$$\pi E_p (I \Leftrightarrow Q) = \gamma_p \pi_p e_{p+1}$$

$$p < m < w \Rightarrow \pi E_m (I \Leftrightarrow Q) = \gamma_m \pi_m (e_{m+1} \Leftrightarrow e_m)$$

(e_m étant le $(m+1)^{ime}$ vecteur de la base canonique, la premiére position étant indéxée par 0)

On en déduit que:

$$\begin{bmatrix} \pi_0 & \cdots & \cdots & \pi_w \\ & \pi_1 & \cdots & \pi_w \\ & & \ddots & \vdots \\ (0) & & & \pi_w \end{bmatrix} (I \Leftrightarrow Q) = \left[\begin{array}{cccc|c} \pi_0 & \cdots & \cdots & \pi_p & \\ & \pi_1 & \cdots & \pi_p & (0) \\ & & \ddots & \vdots & \\ (0) & & & \pi_p & \\ \hline & & & & \begin{array}{cccc} \gamma_p\pi_p & & & (0) \\ \Leftrightarrow\gamma_{p+1}\pi_{p+1} & \gamma_{p+1}\pi_{p+1} & & \\ & \ddots & \ddots & \\ (0) & \Leftrightarrow\gamma_{w-1}\pi_{w-1} & & \gamma_{w-1}\pi_{w-1} \end{array} \\ (0) & & & \end{array}\right]$$

On définit alors les $(w+1) \times (w+1) \Leftrightarrow$matrices suivantes:

$$D_1 = \begin{bmatrix} \pi_0 & & (0) \\ & \ddots & \\ (0) & & \pi_w \end{bmatrix} ; D_2 = \begin{bmatrix} 1 & & & & (0) \\ & \ddots & & & \\ & & 1 & & \\ & & \gamma_p\pi_p & & \\ & & & \ddots & \\ (0) & & & & \gamma_{w-1}\pi_{w-1} \end{bmatrix} ; D_3 = \begin{bmatrix} \pi_0 & & & & (0) \\ & \ddots & & & \\ & & \pi_p & & \\ & & 1 & & \\ & & & \ddots & \\ (0) & & & & 1 \end{bmatrix} ;$$

$$R = \begin{bmatrix} 1 & \dots & 1 \\ & \ddots & \vdots \\ (0) & & 1 \end{bmatrix} ; X = \left[\begin{array}{ccc|ccc} 1 & \dots & 1 & & & \\ & \ddots & \vdots & & (0) & \\ (0) & & 1 & & & \\ \hline & & & 1 & & (0) \\ & (0) & & \Leftrightarrow 1 & \ddots & \\ & & & & \ddots & \ddots \\ & & & & & \Leftrightarrow 1 \quad 1 \end{array}\right]$$

Et l'on peut écrire le résultat sous la forme $RD_1(I \Leftrightarrow Q) = D_2 X D_3$. On en déduit que:

$$(I \Leftrightarrow Q)^{-1} = D_3^{-1} X^{-1} D_2^{-1} R D_1$$

L'inverse d'une matrice diagonale se calcule trivialement. Pour ce qui est de l'inverse de X, on trouve facilement que:

$$X^{-1} = \left[\begin{array}{ccc|ccc} 1 & \Leftrightarrow 1 & & (0) & & \\ & \ddots & \ddots & & (0) & \\ & & \ddots & \Leftrightarrow 1 & & \\ (0) & & & 1 & & \\ \hline & & & & 1 & (0) \\ & (0) & & & \vdots \quad \ddots & \\ & & & & 1 \quad \dots \quad 1 \end{array}\right]$$

On trouve alors que:

$$\begin{aligned} E(\bar{N}) &= \pi\left((I \Leftrightarrow Q)^{-1} \Leftrightarrow I\right)\bar{1} = \pi(D_3^{-1} X^{-1} D_2^{-1} R D_1 \Leftrightarrow I)\bar{1} \\ &= \sum_{u=p}^{w-1} \frac{\left(\sum_{v=u+1}^{w} \pi_v\right)^2}{\gamma_u \pi_u} = \sum_{u=p}^{w-1} \frac{\left(1 \Leftrightarrow \sum_{v=0}^{u} \pi_v\right)^2}{\gamma_u \pi_u} \end{aligned}$$

Ce nombre est $x = \sum_{u=0}^{p} \pi_u \sum_{u=p}^{w-1} \frac{\left(1-\sum_{v=0}^{u} \pi_v\right)^2}{\gamma_u \pi_u}$ fois supérieur au nombre moyen $\frac{1}{\sum_{u=0}^{p} \pi_u}$ d'itérations de l'algorithme qui effectue des pivots indépendants.

En particulier, si $p = 0$ (version la plus simple du décodage par ensemble d'information), ou si p est petit devant w, on peut approximer x par $\frac{1}{\gamma_p} = \frac{k(n-k)}{(k-p)(w-p)} > \frac{n-k}{w}$.

Le coût total des pivots est donc divisé par $\frac{nR(1-R)}{x}$, tandis que le coût total des autres opérations est multiplié par x. N'effectuer que des pivots vicinaux n'est donc intéressant que si le coût d'une itération hors pivot est inférieur à $\frac{1-\frac{nR(1-R)}{x}}{x-1}$ fois le coût d'un pivot, ce qui n'est pas le cas, pour n suffisamment grand, pour les algorithmes de décodage mixte paramétrés de manière optimale (c.a.d. les algorithmes les plus rapides), puisque l'on a vu qu'alors chaque itération avait un coût exponentiel.

Cette dernière trouvaille etait en quelque sorte la raison d'être de cette section. On retiendra, de manière informelle, qu'en effectuant des pivots vicinaux plutôt qu'indépendants, on augmente le nombre d'itérations par un facteur d'environ $\frac{n-k}{w}$.

2.4.3 Cas où le coset leader n'est pas unique

Comme je l'ai déjà dit, le cas où le coset leader n'est pas unique est très marginal, et même si les résultats mentionnés jusqu'ici sont inexacts dans ce cas, cela n'influe pas sur les performances des algorithmes. Il est donc justifié de le négliger.

Cependant, on voit souvent les résultats de la section précédente adaptés au problème de la recherche de mots de poids minimal dans un code (problème proche de celui du décodage), pour lequel la pluralité des solutions est monnaie courante, la plupart des codes étudiés ayant un nombre important de mots de poids minimal.

Dans ce cas, les résultats peuvent varier de manière importante. On pourra facilement adapter l'étude qui suit au problème de la recherche de mots de poids minimal et réparer les éventuelles erreurs obtenues habituellement en négligeant la pluralité des solutions.

Supposons donc que l'on ait m cosets distincts de poids minimal (ce qui signifie en particulier qu'il sera difficile de reconnaître le motif d'erreur et qu'une erreur de décodage aura sûrement lieu, mais oublions quelques instants notre premier objectif) et que l'on attend juste de l'algorithme qu'il trouve l'un quelconque d'entre eux (nous appellerons encore $\bar{N}$ le numéro de la première itération pendant laquelle un coset leader sera calculé).

On associe à chaque itération i de l'algorithme non pas une, mais m variables aléatoires $U_i^j \in \{0...w\}; j = 1...m$, chacune étant associée à l'un des cosets leaders et représentant le nombre de ses 1 dans l'ensemble d'infomation.

L'hypothèse que nous allons faire est incorrecte mais vaut mieux que de considérer que le coset leader est unique: Nous allons supposer que ces m variables aléatoires sont identiques et indépendantes.

On peut définir la matrice Q et faire la même étude que précédemment et l'on trouvera encore que pour tout $1 \leq j \leq m$:

$$\Pr(\forall i \leq N \quad E(U_i^j)) = \pi Q^{N+1} \bar{1}$$

On en déduit que:

$$\Pr(\bar{N} > N) = \Pr(\forall i \leq N \quad \forall j = 0...m \quad E(U_i^j)) = \left(\pi Q^{N+1} \bar{1}\right)^m$$

Et encore que:

$$E(\bar{N}) = \sum_{N=0}^{\infty} \left(\pi Q^{N+1} \bar{1}\right)^m$$

On ne pourra pas dans ce cas, utiliser une inversion de matrice pour déterminer l'espérance du nombre d'itérations.

Conclusion

Ce chapitre permettra en principe d'implémenter ce qui se fait de mieux à ce jour en matière d'algorithme de décodage dur quasi-complet et général (prenant en entrée une matrice génératrice du code, pouvant donc décoder n'importe quel code).

Nous avons jeté quelque lumière sur l'ordre exponentiel de la complexité des différents algorithmes, et l'on se rend bien compte avec le dernier algorithme que faire baisser davantage cet ordre exponentiel demanderait un effort conceptuel important et que le polynôme multipliant le terme exponentiel serait sans doute gigantesque.

Par ailleurs, cet historique de l'ordre exponentiel suggère que l'on se rapproche d'une limite indépassable. Un tel résultat ne saurait être démontré facilement car l'existence d'une telle limite strictement positive impliquerait une preuve de $P \neq NP$.

Si l'on considère une approximation plus précise de la complexité, où le polynôme n'est plus négligé, il apparaît que le décodage mixte utilisant des codes poinçonnés est le plus rapide pour une vaste gamme de longueur et de dimension. Il apparaît également, et cela surprendra davantage les connaisseurs, que n'effectuer que des pivots vicinaux augmente en général la complexité.

Ces algorithmes peuvent avoir d'autres applications que le décodage. Ce chapitre servira par exemple de référence dans la deuxième partie pour la mise au point de ce que nous appellerons les générateurs de mots de poids faible. On considèrera un algorithme de décodage dur sans critère d'arrêt (ne s'arrêtant donc pas, et tournant sur un processeur parallèle par exemple), explorant le coset du mot nul (le code lui-même) et générant des mots de poids faible différents du mot nul.

C'est aussi de cette manière que l'on cherche des mots de poids minimal dans un code (on a cette fois un critère d'arrêt)... Selon les applications toutefois, le nombre maximal d'itérations et les différents paramètres des algorithmes doivent être adaptés spécifiquement.

Chapitre 3

Décodage souple

La patience est l'art d'espérer.
Marquis de Vauvenargues (Réflexions et maximes)

A trois cent quatre-vingts kilomètres à l'heure, il sentit qu'il approchait de la vitesse maximale de son vol en palier. A quatre cents, il estima qu'il était impossible d'aller plus vite. Il en fut un peu chagrin. Il y avait donc une limite [...]

La bonne méthode, mon cher Fletcher, consiste à n'essayer de transcender nos limites que l'une après l'autre, avec patience...
Richard Bach (jonathan livingstone le goéland)

Les faits ne pénètrent pas dans le monde où vivent nos croyances, ils n'ont pas fait naître celles-ci, ils ne les détruisent pas ; ils peuvent leur infliger les plus constants démentis sans les affaiblir...
Marcel Proust (Du côté de chez Swann)

Introduction

Dans l'optique d'optimiser les paramètres d'une transmission, il importe de limiter au maximum les pertes d'information pouvant survenir à chaque phase du processus.

L'une de ces phases, intervenant au début du processus de réception, consiste à mesurer le signal physique porteur d'information. Selon la manière dont elle est mise en oeuvre, une perte d'information plus ou moins grande a lieu.

Ainsi, un démodulateur «dur» compare ce signal à certains seuils et prend une décision quant au «symbôle» reçu en fonction du résultat de ces comparaisons. On dit alors que l'alphabet de sortie du canal est le même que l'alphabet

d'entrée. Mais ce faisant, la valeur précise réellement mesurée du signal est perdue, ce qui constitue une perte d'information.

Un démodulateur souple, quant à lui, ne prend pas de décision et fournit à la couche physique qui lui succède une valeur représentant le plus fidèlement possible ce qu'il a mesuré, minimisant ainsi la perte d'information. L'alphabet de sortie du canal est alors plus grand que l'alphabet d'entrée; dans un cas idéal, cet alphabet a la puissance du continû; en pratique il aura rarement plus d'une centaine d'éléments distincts.

Une connaissance statistique du canal permet alors, à partir de cette valeur, de calculer les probabilités pour que le symbôle émis soit tel symbôle ou tel autre. Il importe de tirer le meilleur parti de cette information supplémentaire. Telle est l'ambition du décodage souple à maximum de vraisemblance.

Pour un canal gaussien, on a vu que pour obtenir une capacité donnée, le rapport signal à bruit nécessaire est inférieur d'environ 1.4 dB si la sortie du canal est réelle plutôt que binaire. En pratique, le gain obtenu par rapport au cas «dur» est souvent estimé à 2 decibels. Dans certains cas [DK00], on peut même trouver une valeur précise pour ce gain, à savoir $10 \log_{10} \pi/2$ dB (≈ 2 dB), c'est à dire qu'un rapport signal à bruit $\pi/2$ fois moins important dans le cas souple que dans le cas dur permet d'obtenir la même qualité de transmission; ou encore, si l'on ne dispose pas d'un démodulateur souple, il faut une énergie de 57% supérieure pour obtenir la même qualité. A notre époque où chaque fraction de décibel gagnée est un enjeu fort disputé, ce gain est colossal et il existe un important besoin en algorithmes de décodage souple.

3.1 contexte et notations

Les notations qui seront employées dans l'étude du cas souple se doivent d'être adaptées au caractère «continu» de l'information souple et des probabilités qu'elle transporte.

Il serait hors de propos ici de s'en tenir rigoureusement au formalisme de la théorie axiomatique des probabilités et de définir des pages de notations, et semble nettement plus intéressant, tant du point de vue de l'écriture que de celui de la lecture et sans que cela nuise à la rigueur des raisonnements, d'utiliser un formalisme répondant à l'intuition.

Ainsi, nous nous autoriserons certains abus de langage et parlerons par exemple de «fonctions» alors que le terme le plus approprié serait sans doute celui de «distributions». Ou bien encore, si x et y sont deux réels, nous confondrons $x < y$ et $x \leq y$ puisque l'évènement $x = y$ a en général, dans le cas continu, une probabilité (mesure) nulle.

Nous nous fierons aux faits que l'intuition permettra de surmonter les problèmes que ces abus pourraient poser et que la simplicité nécessaire des modèles que nous nous donnons éloignera de toute façon ces problèmes, rendant inutile un formalisme trop lourd.

Lorsqu'une variable aléatoire X réelle et continue (il en sera ainsi de celles que nous manipulerons) possède une densité de probabilité donnée (distribution

de $\mathbf{R}$ dans $\mathbf{R}^+$, intégrable, d'intègrale égale à 1), nous noterons (pour tout réel x) $\Pr(X < x)$ la probabilité pour qu'une réalisation de X soit inférieure à x, et $\rho(X = x)$ la densité de probabilité en x qui est la dérivée de $x \rightarrow \Pr(X < x)$. Et nous utiliserons, aussi bien pour la notation «Pr» que pour la notation «ρ», la barre verticale pour les probabilités conditionnelles.

La «fonction» $\rho(X = x)$ correspond intuitivement à la probabilité pour qu'une réalisation de X soit égale à x ou plus précisément à la relation $\Pr(X \in [x, x + dx]) = \rho(X = x)dx$. Elle trouvera donc sa place légitime dans une intégrale, et autorise les opérations traditionnelles concernant la prise en compte de «conditions» (probabilité pour que ... sachant que ...).

3.1.1 Canal, modulation et alphabet

Nous supposerons que le canal est sans mémoire et à bruit blanc, gaussien et additif de densité $N0$, c'est-à-dire que la différence ϵ entre le signal émis et le signal reçu (nous appellerons cette différence le «bruit») dans l'espace des phases est une variable aléatoire complexe de densité:

$$\rho(\epsilon = e) = \frac{1}{\pi N0} \exp\left(\Leftrightarrow \frac{|e|^2}{N0}\right) \qquad e \in \mathbf{C}$$

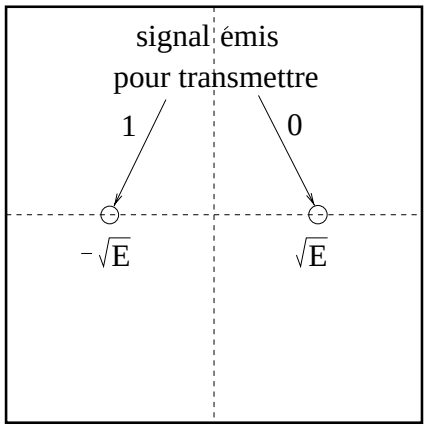


FIG. 3.1: *Modulation antipodal*

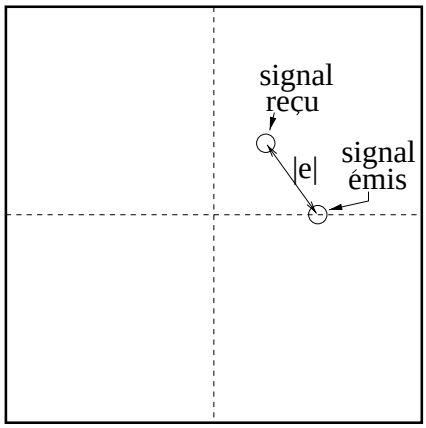


FIG. 3.2: *Bruit additif*

Nous supposerons qu'une modulation antipodal (Binary Phase Shift Keying) d'énergie $\mathcal{E}^2$ est utilisée. C'est à dire que pour chaque bit, l'émetteur envoie un signal d'amplitude $\mathcal{E}$ et de phase 0 ou π selon que le bit vaut (respectivement) 0 ou 1. Nous dirons que le bit $b \in \{0, 1\}$ est envoyé si le signal $(\Leftrightarrow 1)^b\ \mathcal{E}$ est émis. Pour une telle modulation, seule la partie réelle, ou abscisse, du signal reçu dans l'espace des phases porte l'information.

Il serait équivalent de considérer une modulation de phase à quatre états (Quaternary Phase Shift Keying). En effet, puisque quatre signaux distincts

peuvent être transmis, chaque signal porte en lui 2 bits d'information. On peut par exemple supposer que l'abscisse du signal porte le premier bit et que l'ordonnée, porte le second. L'abscisse et l'ordonnée du bruit étant statistiquement indépendante, cela est exactement équivalent à deux modulations antipodales. Et toute théorie s'appliquant à l'un des deux systèmes peut être adaptée à l'autre.

Puisque nous considérons une modulation antipodale, nous projèterons systématiquement l'espace des phases sur l'axe des abscisses, et le signal pourra être représenté par un nombre réel. De même, le bruit sera une variable aléatoire réelle, gaussienne, centrée, et de variance $N0/2$ ($N0/2$ est aussi appelée densité «monolatérale» de bruit):

$$\rho(\epsilon = e) = \frac{1}{\sqrt{\pi N0}} \exp\left(\Leftrightarrow \frac{e^2}{N0}\right) \qquad e \in \mathbf{R}$$

En l'absence d'ambiguïté, nous confondrons un bit et le nombre réel $\pm\mathcal{E}$ le représentant dans l'espace des phases; de même nous évoquerons indifféremment un mot binaire de longueur n et un élément de $\{\Leftrightarrow\mathcal{E}, \mathcal{E}\}^n$.

Cette fois, les mots reçus ne sont plus binaires. L'alphabet de sortie du canal est plus grand. Dans le cas idéal, il s'agit de l'ensemble des nombres réels; en réalité, le démodulateur ne peut généralement pas distinguer plus de 256 régions dans l'espace des phases, mais cela n'a guère d'importance puisque rien que 8 régions distinctes assurent déjà des performances très proches de celles qui seraient rendues possibles par un alphabet continu.

Nous supposerons donc que le démodulateur mesure et envoie au décodeur un nombre réel: l'abscisse du signal.

3.1.2 Variables aléatoires

Nous considérerons les variables aléatoires suivantes:

- le bit transmis, élément de $\mathbf{F}_2$, sera représenté par une variable aléatoire β. Il vaut 0 ou 1 avec une probabilité 1/2.

- Le mot transmis, élément du code C, est une variable aléatoire γ. Par hypothèse, la distribution est homogène, c.a.d. que tous les élément de C ont la même probabilité d'être transmis:

$$\forall c \in C \qquad \Pr(\gamma = c) = 2^{-k}$$

- Le bruit, nombre réel, que l'on a déjà mentionné, est une variable aléatoire continue ϵ dont la densité de probabilité dépend du canal. Pour le canal sans mémoire, à bruit blanc, gaussien et additif de densité monolatérale $N0/2$, nous avons:

$$\forall x \in \mathbf{R} \qquad \rho(\epsilon = x) = \frac{1}{\sqrt{\pi N0}} \exp\left(\Leftrightarrow \frac{x^2}{N0}\right)$$

– L'erreur de transmission, $n\Leftrightarrow$uplet de nombres réels, est le produit combinatoire de n variables aléatoires ϵ représenté par la variable aléatoire δ. Sa densité de probabilité est:

$$\forall d \in \mathbf{R}^n \qquad \rho(\delta = d) = (\pi N0)^{-n/2} e^{-\frac{\sum_{i=1}^{n} d_i^2}{N0}}$$

– Le signal «reçu», nombre réel, somme du signal transmis ($\pm\,\mathcal{E}$) et du bruit, est une variable aléatoire α. Sa densité de probabilité est définie par celles de β et ϵ. C'est le réel qui est mesuré par le démodulateur.

– Le mot «reçu», $n\Leftrightarrow$uplet de nombres réels, somme du mot transmis modulé et de l'erreur de transmission, est une variable aléatoire λ. Sa densité de probabilité est définie par celle de γ et δ.

L'observation dont disposera l'algorithme de décodage pour prendre sa décision sera une réalisation y de la variable aléatoire λ. Le maximum de vraisemblance est atteint par le mot de code $\tilde{c}$ qui maximise la quantité $\Pr(\gamma = \tilde{c}, \lambda = y) = \Pr(\gamma = \tilde{c}) \Pr(\lambda = y | \gamma = \tilde{c}) = 2^{-k} \Pr(\lambda = y | \gamma = \tilde{c})$.

3.1.3 Calcul des probabilités, maximum de vraisemblance et distance généralisée

Pour $b \in \mathbf{F}_2, c = (c_1 ... c_n) \in C$ et $a \in \mathbf{R}, y = (y_1 ... y_n) \in \mathbf{R}^n$, nous noterons $\Pr(\beta = b | \alpha = a)$ la probabilité pour que le bit envoyé vaille b si le réel a est mesuré et $\Pr(\gamma = c | \lambda = y)$ la probabilité pour que le mot envoyé vaille c si le mot $y = (y_1 ... y_n)$ est reçu.

$$\Pr(\beta = b | \alpha = a) = \Pr(\epsilon = a \Leftrightarrow (\Leftrightarrow 1)^b \, \mathcal{E} \, | \alpha = a)$$
$$= \frac{e^{\frac{-(a-(-1)^b \, \mathcal{E})^2}{N0}}}{e^{-\frac{(a-\mathcal{E})^2}{N0}} + e^{-\frac{(a+\mathcal{E})^2}{N0}}} = \frac{1}{1 + e^{\frac{-(-1)^b 4a \, \mathcal{E}}{N0}}} = \frac{e^{\frac{2(-1)^b a \, \mathcal{E}}{N0}}}{2\cosh(\frac{2a \, \mathcal{E}}{N0})}$$

Pour alléger les notations, nous définissons la quantité $A = \frac{N0}{2\,\mathcal{E}}$. Ainsi, nous avons:

$$\Pr(\beta = 0 | \alpha = a) = \frac{e^{a/A}}{2\cosh(a/A)} \qquad \Pr(\beta = 1 | \alpha = a) = \frac{e^{-a/A}}{2\cosh(a/A)}$$

Si a est positif (resp. négatif), β vaut plus probablement 0 (resp. 1), nous définissons donc la fonction b de R dans $\{0,1\}$ par $(b(a) = 0) \Leftrightarrow (a > 0)$. La quantité $|a|$ s'appelle fiablité de b(a).

Le décodeur reçoit donc en entrée un vecteur $y = (y_1 ... y_n) \in R^n$.

Décoder à maximum de vraisemblance consiste à trouver, connaissant y, le mot de code le plus probablement envoyé. Si tous les mots de code sont a priori équiprobables et si le canal est sans mémoire, la probabilité $\Pr(\gamma = c | \lambda = y)$ pour qu'un mot de code $c = (c_1 ... c_n)$ ait été envoyé sachant que y a été mesuré

est proportionnelle à $\prod_{i=1}^{n} \Pr(\beta = c_i | \alpha = y_i)$, et la somme sur tous les mots du code de ces probabilités vaut nécessairement 1.

On a donc:

$$\forall c \in C \qquad \Pr(\gamma = c | \lambda = y) = \frac{e^{\sum_{i=1}^{n} (-1)^{c_i} y_i / A}}{\sum_{c' \in C} e^{\sum_{i=1}^{n} (-1)^{c'_i} a_i / A}}$$

Maximiser (sur C) cette probabilité revient donc à maximiser la quantité $\sum_{i=1}^{n} (\Leftrightarrow 1)^{c_i} y_i$ souvent appelée corrélation de y et c.

Soit $m = (b(y_1)...b(y_n)) \in \{0,1\}^n$ le mot qui serait renvoyé par un démodulateur «dur». m peut également être vu comme le mot qui maximise (sur tout l'espace ambiant) la corrélation avec y, laquelle vaut $\sum_{i=1}^{n} |y_i|$.

La corrélation de y et de tout mot de code c vaut donc:

$$\sum_{i=1}^{n} |y_i| \Leftrightarrow 2 \sum_{i=1}^{n} (c_i \oplus m_i) |y_i|$$

(où $\oplus$ représente l'addition modulo 2).

En décodage dur, les motifs d'erreur possibles étaient les mots $e = m \oplus c$ ($c \in C$), et il s'agissait d'en minimiser le poids de Hamming. En décodage souple, il s'agira de minimiser la quantité $\sum_{i=1}^{n} (c_i \oplus m_i) |y_i|$ que nous définissons ci-dessous comme étant le poids généralisé du motif d'erreur.

Définition 10 *Nous appellerons poids généralisé de* $e = (e_1...e_n) \in \{0,1\}^n$ *et nous noterons* $w_G(e)$ *la quantité* $\sum_{i=1}^{n} e_i |y_i|$. *Pour tout sous-ensemble* I *du support, nous appellerons poids généralisé sur* I *de* e *et nous noterons* $w_G^I(e)$, *la quantité* $\sum_{i \in I} e_i |y_i|$.

Ce poids généralisé est aussi parfois appelé métrique ellipsoïdale (l'analogie avec les ellipsoïdes de l'espace euclidien est évidente si l'on note $(c_i \Leftrightarrow m_i)^2$ la quantité $c_i \oplus m_i$). Les quantités $|y_i|$ déterminant la forme des ellipsoïdes de $\mathbf{F}_2^n$ définis par un poids généralisé constant. Pour des $|y_i|$ tous égaux par exemple, ces ellipsoïdes sont des sphères de Hamming.

Ainsi, alors que les algorithmes de décodage dur cherchait l'élément du coset de m de plus petit poids de Hamming, les algorithmes de décodage souple devront, quant à eux, chercher l'élément du coset de m de plus faible poids généralisé.

Cette différence rend le décodage souple plus précis. C'est-à-dire que pour un canal analogue, le taux d'erreur résiduel (resp. le gain de codage) du décodage souple à maximum de vraisemblance est inférieur (resp. supérieur) à celui du décodage dur complet.

3.2 Outils pour le décodage spécifiques au décodage souple

3.2.1 Critère d'arrêt

Rappelons que dans le décodage dur, il est parfois possible en examinant un élément du coset du mot reçu, de décider avec certitude qu'il s'agit du coset leader. On sait en effet que tout mot de poids inférieur ou égal au rayon d'empilement (capacité de correction) du code est nécessairement de poids minimal dans son coset.

Le coset leader peut cependant être de poids strictement supérieur au rayon d'empilement et l'on ne peut alors avoir aucune certitude sur simple observation de ce mot quant à la minimalité de son poids.

La comparaison du poids au rayon d'empilement est tout de même un critère d'arrêt largement répandu pour les algorithmes de décodage dur, et il serait intéressant qu'il existe un critère analogue pour le décodage souple et sa métrique ellipsoïdale.

G. David Forney, Jr, dans son article fondateur sur le décodage souple [Jr66] de 1966, n'utilisait pas tout à fait la même métrique que nous. Pour décrire sa métrique en nous ramenant à nos notations définissons pour tout i, $z_i = \min(|y_i|, 1)$ et $w_F(e) = \sum_{i=1}^{n} e_i z_i$. Le but de son algorithme était de trouver le mot de code c qui minimise la quantité $w_F(m \oplus c)$ (m étant le mot de décision dure, ou mot reçu, défini en section précédente), et il a démontré que si d est la distance de Hamming minimale du code, alors:

$$\forall c \in C \quad \sum_{i=1}^{n} (\Leftrightarrow 1)^{c_i \oplus m_i} z_i > n \Leftrightarrow d \Rightarrow \forall c' \in C \Leftrightarrow \{c\} \quad w_F(m \oplus c') > w_F(m \oplus c)$$

Son critère d'arrêt consistait donc, observant un élément $m \oplus c$ du coset, à le considérer comme étant le coset leader si $\sum_{i=1}^{n} (\Leftrightarrow 1)^{c_i \oplus m_i} z_i$ était supérieur à $n \Leftrightarrow d$.

En 1980, Daniel J. Costello, Jr. et Christopher Chi-Hsun Yu [YJ80] améliorent ce critère en considérant non pas $z_i = \min(|y_i|, 1)$ (qui dans une optique «maximum de vraisemblance» est sous-optimal) mais $z_i = \frac{|y_i|}{\max_{j=1\ldots n} |y_j|}$ de manière à avoir toujours $0 \leq z_i \leq 1$ mais sans biaiser les probabilités, le critère d'arrêt ayant ensuite la même forme que celui de Forney. Les résultats ne peuvent qu'être meilleurs car minimiser la métrique ainsi définie est équivalent à décoder à maximum de vraisemblance (la métrique ainsi définie est proportionnelle au poids généralisé w_G défini en section précédente).

Mais, de même qu'en décodage dur le poids du coset leader n'est pas toujours inférieur au rayon d'empilement, les critères d'arrêt définis ci-dessus ne permettront pas toujours de capturer le motif d'erreur le plus vraisemblable. Il importe donc que le critère capture le maximum de motifs. C'est de ce point de vue que le critère est encore amélioré en 1991 par Dana J. Taipale et Michael B. Pursley [TP91].

Pour tout élément $m \oplus c$ du coset, de poids de Hamming inférieur à la distance minimale d du code, soit f le mot de l'espace ambiant tel que: $\mathrm{wt}(f) = d \Leftrightarrow \mathrm{wt}(m \oplus c)$; $\mathrm{supp}(m \oplus c) \cap \mathrm{supp}(f) = \emptyset$ et composé des positions les plus fiables en dehors du support de $m \oplus c$. Taipale montre alors que:

$$w_G(m \oplus c) < w_G(f) \Rightarrow \forall c' \in C \Leftrightarrow \{c\} \quad w_G(m \oplus c') > w_G(m \oplus c)$$

Ce qui définit un critère d'arrêt. Il montre aussi que tout élément du coset satisfaisant le critère de Costello satisfait également le sien mais que le contraire n'est pas vrai. Son critère capturera donc plus souvent le coset leader.

3.2.2 Tri des positions par ordre de fiabilité

Le décodage souple est une généralisation du décodage dur. De certains points de vue, et en particulier de celui de la théorie de la complexité, il s'agit donc d'un problème plus difficile [FCGB88]. Pour cette raison et également parce que sa définition requiert certaines considérations relevant davantage du traitement du signal que des mathématiques, il a longtemps (et encore aujourd'hui) effrayé une part importante des spécialistes du décodage.

On a vu cependant, avec la définition d'une métrique ellipsoïdal, qu'il peut être appréhendé d'un point de vue strictement mathématique. La seule différence avec le décodage dur reposant dans la définition de coefficients de pondérations des positions: ce que l'on a appelé leur fiabilité.

Mais ces coefficients ne rendent pas forcément le problème plus difficile. Au contraire dans certains cas ils peuvent permettre aux algorithmes de trouver la solution plus rapidement, en leur fournissant un ordre pertinent pour mener leurs explorations.

L'algorithme de recherche de motifs d'erreur défini en section 2.1, par exemple, génèrera les différents motifs d'erreur par poids généralisé croissant plutôt que par poids de Hamming croissant (jusqu'à ce que l'un d'eux ait le même syndrôme que le mot reçu).

L'information souple (la fiabilité des bits) permet donc d'ordonner de manière plus précise les différentes possibilités par rapport à leur vraisemblance, ce qui constitue un avantage important. Mais cela est encore plus flagrant pour le décodage par ensemble d'information. Nous y reviendrons plus longuement mais considérons en attendant la première amélioration de l'algorithme précédent: La restriction à un ensemble d'information.

Plutôt que de générer les n bits d'un nouveau motif, on génère les k bits d'un nouveau motif restreint à un ensemble d'information donné, les $n \Leftrightarrow k$ bits restants étant déterminés de manière à ce que le motif ainsi généré appartienne au coset du mot reçu.

L'algorithme génèrera le bon motif d'erreur dès que le motif restreint testé correspondra aux erreurs de transmission sur l'ensemble d'information considéré. Prenons alors comme ensemble d'information celui qui est constitué des k bits les plus fiables (si ces k bits ne constituent pas un ensemble d'information, on pourra en générer un sous-optimal grâce à un algorithme glouton par

exemple), que nous appellerons la «base la plus fiable», et générons les différents motifs (restreints) par poids généralisé croissant.

Il paraît clair, compte tenu de la défnition probabilistique de la fiablité, que l'espérance du nombre d'erreur de transmission sur un ensemble de k positions est minimal si cet ensemble est constitué des k bits les plus fiables. L'algorithme tombera donc plus tôt en moyenne sur le bon motif s'il choisit la base la plus fiable plutôt qu'un ensemble d'information quelconque. L'information souple permet par conséquent d'accélérer cet algorithme.

L'algorithme ainsi défini a été proposé pour la première fois par B. G. Dorsch en 1974 [Dor74] avec un critère d'arrêt simple et astucieux. Plus de 20 ans plus tard, Marc P. C. Fossorier et Shu Lin [FL95] mènent une étude probabiliste assez poussée de cet algorithme (qu'ils réinventent pour l'occasion, ignorant l'infortuné Dorsch). Il a été repris (le mot n'est peut-être pas approprié, la correspondance de Dorsch n'étant toujours pas citée) et amélioré en 1997 par D. Gazelle et J. Snyders [GS97] qui utilisent la distribution des poids du code pour définir un critère d'arrêt complexe mais plus efficace que celui de Dorsch.

Des améliorations successives voient ensuite le jour [FL96a] [FL97] [FL99], semblant refléter le fait qu'un principe plus puissant doit exister. Et pour cause, on a vu dans le chapitre traitant du décodage dur qu'il était plus intéressant de traiter un nombre limité de candidats sur un grand nombre d'ensembles d'information différents que le contraire. Nous essayons dans la prochaine section d'adapter ce principe au décodage souple. Mais voyons déjà ce que l'on peut dire de la statistique du bruit sur les positions ordonnées par ordre décroissant de fiabilité.

Suite au tri, les variables aléatoires représentant le bruit affectant les différentes positions ne sont plus identiques et indépendantes. Supposons le tri effectué, et appelons ϵ_i $(i = 1...n)$ la variable aléatoire représentant le bruit affectant la $i\Leftrightarrow$ième position, et β_i $(= \pm\mathcal{E})$ le $i\Leftrightarrow$ème signal transmis.

Fossorier a montré [FL95] que l'on a les distributions de probabilités suivantes, pour $i = 1...n$ (Q est la fonction $x \mapsto \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{t^2}{2}} dt$, et $\tilde{Q}(x) = Q\left(\sqrt{\frac{2\,\mathcal{E}^2}{N0}} x\right)$):

$$\forall x \in R \qquad \rho\left(\frac{\epsilon_i}{\beta_i} = x\right) =$$

$$i\binom{n}{i} \frac{\mathcal{E}\, e^{-x^2 \frac{\mathcal{E}^2}{N0}}}{\sqrt{\pi N0}} (\tilde{Q}(|x{+}1|\Leftrightarrow 1){+}\tilde{Q}(|x{+}1|{+}1))^{i-1} (\tilde{Q}(1\Leftrightarrow|x{+}1|)\Leftrightarrow\tilde{Q}(1{+}|x{+}1|))^{n-i}$$

Définissons $P_e^i(n)$ la probabilité pour que la $i\Leftrightarrow$ième position soit erronée, on a bien-sûr $P_e^i(n) = \int_{-\infty}^{-1} \rho\left(\frac{\epsilon_i}{\beta_i} = x\right) dx$.

On trouvera les résultats suivants démontrés dans [BC91]:

Proposition 9

$$\forall i = 1...n \Leftrightarrow 1 \quad P_e^i(n) < P_e^i(n \Leftrightarrow 1) < P_e^{i+1}(n)$$

$$P_e^i(n) = \sum_{l=n-i+1}^{n} (\Leftrightarrow 1)^{l-n+i-1} \binom{n}{l} \binom{l \Leftrightarrow 1}{n \Leftrightarrow i} P_e^1(l)$$
$$= \sum_{l=i}^{n} (\Leftrightarrow 1)^{l-i} \binom{n}{l} \binom{l \Leftrightarrow 1}{i \Leftrightarrow 1} P_e^l(l)$$

$$P_e^{i+1}(n) = P_e^i(n) + \binom{n}{i} \sum_{l=0}^{i} (\Leftrightarrow 1)^l \binom{i}{l} P_e^1(n \Leftrightarrow i + l)$$

Et l'on pourra aussi [FL96b] faire l'approximation $P_e^i(n) \approx e^{-4(m_i-1)\frac{\varepsilon^2}{N_0}}$ avec $m_i = \alpha^{-1}(1 \Leftrightarrow i/n)$ et $\alpha(x) = \tilde{Q}(2 \Leftrightarrow x) \Leftrightarrow \tilde{Q}(x)$.

Un grand intérêt de ces probabilités est qu'elles sont définies indépendamment de y, le signal reçu.

3.2.3 Enumération ordonnée des motifs d'erreur

Dans la section précédente, nous avons mis en évidence l'intérêt d'énumérer les différents motifs d'erreur par poids généralisé croissant, en se restreignant éventuellement à un sous-ensemble du support, mais nous n'expliquons pas comment mener à bien cette énumération. Cette section y remédie et s'inspire d'une idée donnée dans [Dum96b] et [Dum01].

Nous donnons ci-dessous une méthode, pour tout entier N, d'énumération des N vecteurs de plus faibles poids généralisé. L'adaptation à un sous-ensemble du support est directe et ne posera aucune difficulté. Cette méthode est similaire aux méthodes conventionnelles de décodage par treillis. Elle consiste en n étapes, à chaque étape j, on construit une liste $X_j = \{(m, w)\}$ de mots m restreints aux sous-ensemble $\{1...j\}$ du support, accompagnés de leur poids généralisé w sur ce sous-ensemble..

On construit X_{j+1} à partir de X_j en concaténant aux mots de cette liste les symbôles 0 ou 1. Si l'on ajoute 0, le poids généralisé ne change pas, si l'on ajoute 1, le poids généralisé est augmenté de $|y_{j+1}|$. Une fois X_{j+1} construite, on peut libérer la mémoire de la liste X_j.

En premier lieu, pour les étapes $j = 1, ..., \lfloor \log_2 N \rfloor$, on construit récursivement les listes complètes X_j composées de 2^j éléments.

Pour les étapes j suivantes, les listes X_j ne comprendront que les N mots de plus faible poids généralisé sur $\{1...j\}$.

Si X_j comprend N mots, la concaténation de 0 et 1 à ces mots donnera $2N$ mots. On les triera par poids généralisé croissant et on ne gardera que les N premiers mots.

On peut faire une économie substantielle en considérant que le symbôle 1 ajouté au $i$ème mot m_i de la liste X_j triée, donnera un mot m en position au moins $2i$ dans la liste X_{j+1} triée. En effet, les $i \Leftrightarrow 1$ mots précédant m_i dans X_j auxquels on ajoute 0 ou 1 donneront $2(i \Leftrightarrow 1)$ mots précédant m dans X_{j+1}, et m_i auquel on ajoute 0 le précèdera aussi. Il sera donc inutile de concaténer le symbôle 1 aux $N/2$ derniers mots de X_j, la liste à trier n'aura que $3N/2$ éléments.

Après la $n$ième étape on a la liste attendue des N mots de plus faible poids généralisé, triée par poids généralisé croissant. Le complexité de cette énumération est majorée par $n\frac{3}{2}N\log\frac{3}{2}N$. Le même type d'énumération pour un sous-ensemble du support de cardinal x aura une complexité majorée par $x\frac{3}{2}N\log\frac{3}{2}N$.

3.3 Décodage par ensemble d'information souple

3.3.1 Problématique

On a vu que le décodage par ensemble d'information offrait un principe puissant pour le décodage dur des codes généraux qui permettait de réduire fortement les complexités. On peut donc s'attendre à ce qu'il en aille de même pour le décodage souple, d'autant que l'information souple devrait pouvoir permettre de choisir pertinemment les ensembles d'information successifs.

On trouvera souvent dans la littérature [FL95] [HHC94] [GS97] l'utilisation de la base la plus fiable comme point de départ. Mais le problème se pose de ce qu'il faut faire ensuite... Cette base est unique et les algorithmes de décodage par ensemble d'information doivent en tester un très grand nombre pour fonctionner correctement.

Une première idée consiste à ordonner l'ensemble des bases par ordre décroissant de fiabilité (la fiabilité d'une base est la somme des fiabilités de ses positions), et à parcourir alors cet ensemble selon cet ordre. Cette possibilité présente cependant des problèmes d'implémentation: Il est en effet plus coûteux de calculer, de stocker et d'ordonner l'ensemble des bases que de décoder, et le faire au fur et à mesure est loin d'être trivial si l'on exige une complexité en mémoire acceptable et un surplus de complexité en temps négligeable.

Par ailleurs, si l'on parvenait à le faire malgré les difficultés d'implémentation et de gestion de la mémoire que cela représente, on explorerait alors des bases voisines les unes des autres et il faudrait attendre longtemps avant de s'éloigner significativement de la base en cours d'exploration, et de rejeter les erreurs qu'elles contient. L'exploration manquerait d'amplitude, tant et si bien que la complexité pourrait augmenter et non diminuer (il n'y a pas à ma connaissance d'étude de la complexité de l'algorithme défini par cette exploration par fiabilité de base décroissante).

Une autre idée consiste alors à choisir les bases successives de manière aléatoire, mais corrélée avec leur fiabilité, c.a.d. que plus une base sera fiable plus elle aura de chance d'être choisie par l'algorithme. Dans la suite nous allons définir deux algorithmes qui utilisent ce principe d'aléa corrélé mais de manière fort différente.

L'un d'entre eux a été introduit par Ilya Dumer à Eurocode 92 [Dum93] et en 96 à Allerton [Dum96a] avec des résultats précis de complexité, malheureusement les preuves des résultats annoncés n'ont pas encore été publiées et ces résultats correspondent à l'étude de la complexité «selon les cas» [1], fort différente

1. selon l'entrée de l'algorithme

de la complexité du décodage quasi-complet.

L'autre est le fruit de mon travail et exploite le phénomène de résonnance stochastique. Il s'agit bien d'un algorithme de décodage quasi-complet, et les simulations témoignent d'un gain en complexité spectaculaire par rapport au décodage dur par ensemble d'information ou par rapport à tous les algorithmes de décodage souple quasi-complet existant «officiellement». Je ne suis malheureusement parvenu à obtenir aucun résultat théorique quant à cette complexité mais une collaboration (post-doc ou « joint paper ») avec Ilya Dumer est prévue et pourrait s'avérer fructueuse.

Si l'on a un algorithme de décodage souple par ensemble d'information, on pourra aisément en déduire des algorithmes de décodage mixte, en s'inspirant de ce qui a été dit à ce propos pour le décodage dur en section 2.3.

On pourra même faire un peu mieux que dans le cas du décodage dur, car on pourra générer les N motifs d'erreur de plus faible poids généralisé sur un ensemble donné de cardinal x, quel que soit N. On pourra donc choisir précisément le nombre N qui optimise la complexité globale du décodage (pour le décodage dur, on génère les motifs d'erreur de poids de Hamming au plus p, soit $\sum_{i=0}^{p} \binom{x}{p}$ motifs, ce qui laisse moins de latitude quant au nombre de motifs, qui doit être une somme de coefficients binomiaux).

3.3.2 Décodage par recouvrement d'ellipsoïdes

Le théorème 7 du chapitre précédent (p. 51) donne, pour tout $w < r < n$, des bornes sur la taille minimale d'un $r\Leftrightarrow$recouvrement de la sphère de Hamming de rayon w sur $\{1...n\}$.

Par ailleurs, il est montré qu'en choisissant aléatoirement, selon une loi uniforme, un nombre de $r\Leftrightarrow$sous-ensembles de même ordre exponentiel que $\frac{\binom{n}{w}}{\binom{r}{w}}$, on réalise un tel recouvrement avec une probabilité arbitrairement proche de 1.

Dumer se propose d'obtenir des résultats similaires pour les ellipsoïdes. Nous exposons dans cette section les principaux résultats qu'il obtient. N'ayant eu accès qu'à une version rapidement écrite et sans démonstration de ces résultats, n'ayant pas pu être parfaitement convaincu par cette présentation et ayant dû corriger quelques coquilles, je ne puis malheureusement pas garantir qu'ils en soient maintenant totalement dépourvus. Il faut tout de même signaler qu'Ilya Dumer travaille encore actuellement sur les interminables démonstrations de ces résultats qui, m'a-t-il confié, lui ont demandé plus de dix ans de recherche.

Soient donc $v_i(= |y_i|) \in \mathbf{R}^+$ la fiabilité du $i$ème bit reçu, et $v = (v_1...v_n)$. On sait que le décodage sera quasi-complet si l'on teste les 2^{n-k} motifs d'erreur les plus probables, c.a.d. les 2^{n-k} mots $(e_1...e_n)$ de $\mathbf{F}_2^n$ minimisant la quantité $\sum_{i=1}^{n} e_i v_i$. Ils forment un ensemble que nous appellerons $E(n, v, 2^{n-k})$.

Puisque, pour tout v' proportionnel à v, le problème est equivalent, on peut, sans perte de généralité, considérer que $\sum_{i=1}^{n} v_i = n$.

Pour tout $\rho \leq n/2$, on définit $\lambda_\rho > 0$ et $p_j(\rho) \leq 1/2$ $(j = 1...n)$ par les

équations suivantes:

$$p_j(\rho) = \frac{1}{1 + 2^{\lambda_\rho v_j}}, \qquad \sum_j p_j(\rho) v_j = \rho$$

Pour tout j, $p_j(\rho)$ croît avec ρ, et λ_ρ décroît avec ρ. On pourra vérifier que $p_j(0) = 0$, $\lambda_0 = \infty$, $p_j(n/2) = 1/2$, $\lambda_{n/2} = 0$.

Soit $\mu < n/2$ tel que $\sum_j H_2(p_j(\mu)) = n \Leftrightarrow k$, et soient $t = \sum_j p_j(\mu)$, et $L = \frac{2^{n-k}}{\binom{n-k}{t}}$.

Théorème 8 *Pour tout $v \in \mathbf{R}^n$ et tout entier $r \leq n$ suffisamment grand, soit $b(n, r, v)$ le cardinal minimal d'un r⇔recouvrement de $E(n, v, 2^{n-k})$. On a, avec les notations des paragraphes précédents:*

$$b(n, r, v) \geq L$$

$$\left(\forall j = 1 \ldots n \quad p_j(\mu) \leq \frac{t}{n \Leftrightarrow k} \right) \Rightarrow b(n, r, v) \stackrel{exp}{=} L$$

Par ailleurs, le théorème suivant est d'une importance majeure:

Théorème 9 *Si l'on choisit aléatoirement $(n \ln n) 2^{\sum_j H_2(p_j(\mu)) \left(1 - H_2\left(\frac{p_j(\mu)}{H_2(p_j(\mu))}\right)\right)}$ sous-ensembles du support de cardinal n⇔k, selon une loi telle que toute position j fasse partie du sous-ensemble choisi avec une probabilité égale à $H_2(p_j(\mu))$, alors ces sous-ensembles recouvrent $E(n, v, 2^{n-k})$ avec une probabilité $1 \Leftrightarrow n^{-n}$.*

La quantité $\sum_j H_2(p_j(\mu)) \left(1 \Leftrightarrow H_2\left(\frac{p_j(\mu)}{H_2(p_j(\mu))}\right)\right)$ est maximale lorsque tous les v_j sont égaux, ce qui équivaut au recouvrement d'une sphère et donc au décodage dur. Elle vaut alors $(n \Leftrightarrow k) \left(1 \Leftrightarrow H_2\left(\frac{H_2^{-1}(1-R)}{1-R}\right)\right)$, et l'on retrouve la complexité du décodage dur. Pour des ellipsoïdes oblongs, la complexité sera strictement inférieure.

Le décodage par recouvrement d'ellipsoïdes est donc tout désigné: Il s'agit d'un décodage par ensemble d'information où les bases successives sont choisies selon une loi satisfaisant la contrainte du théorème ci-dessus.

L'algorithme calculera donc, en fonction de la fiabilité des bits de chaque nouveau mot reçu, les valeurs de $p_j(\mu)$ et $H_2(p_j(\mu))$, ainsi que le nombre d'itérations défini par le théorème.

Chaque itération sera une simple itération de décodage par ensemble d'information (calcul de l'élément du coset s'annulant sur l'ensemble d'information en cours, s'il ne s'agit pas d'un ensemble d'information mais d'un ensemble x⇔défectif, avec $x > 0$, il calculera les 2^x éléments du coset s'anullant sur $k \Leftrightarrow x$ positions indépendantes de cet ensemble).

A chaque itération, on choisit successivement les $n \Leftrightarrow k$ positions du complémentaire de l'ensemble d'information de la manière suivante: on choisit la première position aléatoirement parmi les n positions de manière à ce que chaque

position puisse être choisie avec une probabilité $H_2(p_j(\mu))/(n \Leftrightarrow k)$ (je rappelle que $\sum_j H_2(p_j(\mu)) = n \Leftrightarrow k$).

Pour les $n \Leftrightarrow k \Leftrightarrow 1$ positions suivantes, si J est l'ensemble des positions non encore sélectionnés, on choisit aléatoirement une position dans J de manière à ce que chaque position j de J puisse être choisie avec une probabilité $H_2(p_j(\mu))/\sum_{i\in J} H_2(p_i(\mu))$.

On pourra vérifier qu'ainsi, toute position j fera partie de l'ensemble final avec une probabilité $H_2(p_j(\mu))$. Enfin on considère le complémentaire de cet ensemble comme l'ensemble d'information à traiter.

3.3.3 Décodage par résonnance stochastique

La résonnance stochastique est un phénomène d'aide à la transmission d'un signal par le bruit, ayant lieu dans certains systèmes non-linéaires [MPO94], [WM95], [CB97].

Etonnamment les théoriciens de l'information ne sont pas prévenus, mais d'aucuns physiciens (ignorant manifestement les efforts déployés par d'autres pour s'attaquer à ces problèmes) prétendent depuis quelques années améliorer significativement les rapports signal à bruit de transmissions non périodiques et les capacités des canaux les plus classiques grâce à l'introduction de phénomènes chaotiques ajustables et à des phénomènes de résonnance stochastique [GCB97], [Kis96], [LGK96].

Je préfère ne pas insister sur ce sujet car d'une part, malgré le prestige des journaux où sont parus ces articles, il m'a paru suspect que ces travaux soient dénués de références, ne serait-ce qu'au respectable Shannon, et ne font que se citer mutuellement et parce que d'autre part les phénomènes de résonnance stochastique que j'ai personnellement observés s'appliquent non pas à des phénomènes physiques mais à l'efficacité d'un algorithme.

Je rappelle qu'il s'agit de mettre au point un algorithme de décodage mixte, en choisissant les bases successives de manière aléatoire, mais corrélée avec leur fiabilité, c.a.d. que plus une base sera fiable plus elle aura de chance d'être choisie par l'algorithme.

On choisit donc le décodage mixte avec utilisation de codes poinçonnés (cf. section 2.3.2), qui est le plus rapide pour la plupart des longueurs et dimensions de codes utilisables en pratique. Je rappelle que la longueur s des codes poinçonnés est soumise aux contraintes:

$$k < s < \min(n, 2k); \quad \frac{s}{n} H_2^{-1}\left(2\left(1 \Leftrightarrow \frac{k}{s}\right)\right) < H_2^{-1}(1 \Leftrightarrow R);$$

$$\frac{s}{n}\left(1 \Leftrightarrow H_2^{-1}\left(2\left(1 \Leftrightarrow \frac{k}{s}\right)\right)\right) < 1 \Leftrightarrow H_2^{-1}(1 \Leftrightarrow R).$$

et que, pour le décodage dur, la longueur s optimale minimise:

$$\left(1 \Leftrightarrow \frac{s}{n}\right)\left(1 \Leftrightarrow H_2\left(\frac{nH_2^{-1}(1 \Leftrightarrow R) \Leftrightarrow sH_2^{-1}\left(2\left(1 \Leftrightarrow \frac{k}{s}\right)\right)}{n \Leftrightarrow s}\right)\right)$$

Par ailleurs, il faut générer sur chaque moitié de support des codes poinçonnés, 2^{s-k} motifs d'erreur. Pour générer les 2^{s-k} motifs de plus faible poids généralisé sur une moitié de support poinçonné, on utilisera la technique mentionnée en section 3.2.3.

Reste à définir la manière dont on choisit les bases successives. Voici ce que je suggère.

A chaque itération, considérons $(y_i)_{i=1...n}$ le signal originellement reçu, et pour i allant de 1 à n, calculons, grâce à un générateur pseudo-aléatoire, $y'_i = y_i + x$ où x est une réalisation d'une variable (pseudo-) aléatoire de loi normale, centrée et de variance σ^2 qui est un nouveau paramètre de l'algorithme.

C'est-à-dire qu'à chaque itération, nous ajoutons au signal reçu (de manière non cumulée, on reconsidère le signal original à chaque fois) un bruit gaussien secondaire, créé de toute pièce par l'algorithme.

On choisit alors la base la plus fiable relativement aux y'_i.

L'espérance de x est 0, donc celle de y'_i est y_i. Il est donc clair qu'une position aura d'autant plus de chance de faire partie de la base que sa fiabilité est élevée, et ce, quel que soit σ^2.

Le paramètre σ^2 sert alors à régler l'amplitude de l'exploration. Plus cette variance est faible, plus les bases successives seront proches de la base la plus fiable (chaque itération, considérée indépendamment des autres aura donc une plus grande probabilité de succès) et moins cette exploration aura d'amplitude (les itérations successives sont donc davantage corrélées entre elles, ainsi que leur probabilité de succès).

Nous avons donc deux influences antagonistes du paramètre σ^2, et il faut s'attendre à ce qu'il ait une valeur optimale finie.

Nous avons tracé, sur les figures 3.4 à 3.6, l'influence du paramètre sur le taux d'erreur pour différents nombres N d'itérations, ainsi que la décroissance du taux d'erreur en fonction du nombre d'itérations pour différente valeur du paramètre (sur ces figures, nous avons considéré σ et non σ^2).

Il apparaît qu'effectivement, certaines valeurs du paramètres permettent d'atteindre beaucoup plus rapidement des taux d'erreur proches du taux d'erreur optimal. On dit que pour ces valeurs, on atteint la résonnance stochastique (pour des taux de codage 3/4, ce phénomène est même assez spectaculaire).

J'ai constaté que la valeur optimale pour σ était toujours proche de la moitié de l'écart type de l'erreur, c.a.d. que le σ^2 optimal vaut à peu près le quart de la puissance du bruit. Un résultat aussi simple, accompagné d'une preuve serait évidemment bienvenu, mais mes efforts n'y ont pas suffi.

3.4 Performances

Le nombre d'itérations nécessaire, avec le paramètre optimal, pour que le décodage soit quasi-complet a été évalué pour de nombreuses valeurs de n à taux de codage constant. Il semble être de la forme $K\frac{2^{\alpha n}}{n^2}$. α dépend du rapport signal à bruit et du taux de codage, K dépend en plus du niveau de quasi-complétude requis.

J'ai tracé sur la figure 3.7, les valeurs de α que j'ai estimées (ces calculs demandent beaucoup de simulations) pour trois valeurs de taux de codage et un rapport signal à bruit de 3 dB. On peut constater que cet algorithme surpasse de manière assez nette ses concurrents, non seulement de manière asymptotique, mais surtout pour tout n car le terme exponentiel est à diviser par un polynôme et non à multipler.

L'algorithme, à l'aide de quelques simulations préliminaires ou d'abaques, choisira donc le paramètre σ^2 optimal et le nombre d'itérations adéquat, avant de commencer le décodage.

Conclusion

Le décodage souple quasi-complet et général n'est donc ni plus ni moins qu'un décodage classique avec un alphabet de sortie du canal plus grand que son alphabet d'entrée. Il s'agit simplement de trouver une métrique plus adéquate que la métrique de Hamming.

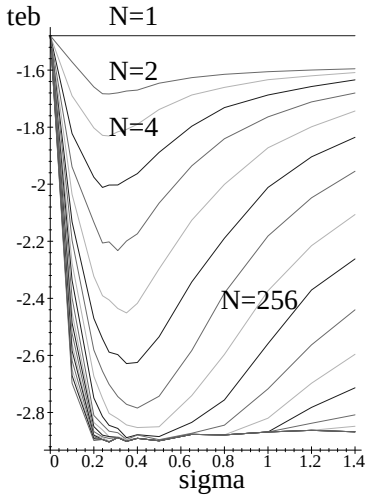
Nous avons traité le cas du canal gaussien à entrée binaire et sortie réelle, et nous avons vu que la métrique optimale a une forme très simple: On pondère chaque position par le module du signal reçu. Décoder (soustraire son coset leader au mot reçu) selon cette métrique permet alors de trouver le maximum de vraisemblance, et d'atteindre le taux d'erreur le plus faible possible.

L'avantage sur le décodage dur est double: le taux d'erreur résiduel est plus faible et le décodage est plus rapide.

Le meilleur algorithme de décodage souple par ensemble d'information (sans recherche de motifs d'erreur) est théoriquement le décodage par recouvrement d'ellipsoïdes d'Ilya Dumer. Le décodage par résonnance stochastique peut s'adapter au décodage par ensemble d'information, mais présente un défaut de résultat théorique, et est sous-optimale par rapport au précédent.

Cependant, il sera difficile d'obtenir des résultats théoriques et des paramètres optimaux pour un décodage mixte (avec recherche de motifs d'erreur), tandis qu'avec son unique paramètres à régler, le décodage par résonnance stochastique s'adapte facilement, et permet d'avoir un décodage mixte quasi-optimal, généralement plus efficace que le décodage par recouvrement d'ellipsoïdes d'Ilya Dumer, basé sur le décodage par ensembles d'information qui est un cas particulier non optimal du décodage mixte.

FIG. 3.3: $n = 128, R = 1/2, E_b/N_0 = 2\,\mathrm{dB}$



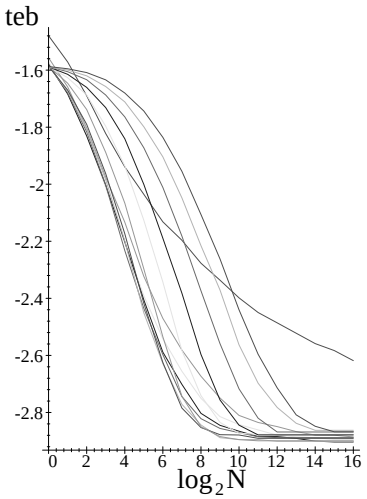
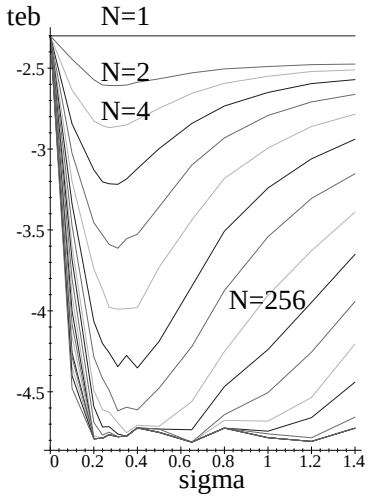
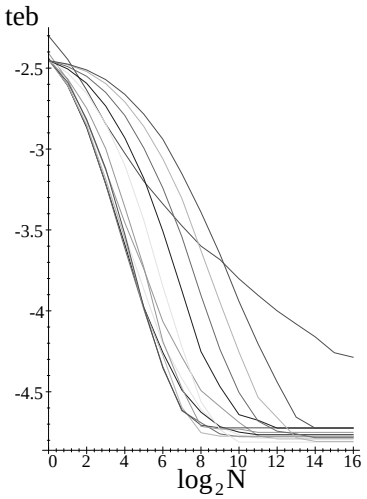


FIG. 3.4: $n = 128, R = 1/2, E_b/N_0 = 3\,\mathrm{dB}$



log_{10} teb en fonction de σ pour (de haut en bas) N =1,2,4,8...



log_{10} teb en fonction de $\log_2 N$ pour les différentes valeurs de σ

FIG. 3.5: $n = 96, R = 1/2, E_b/N_0 = 4\,\mathrm{dB}$

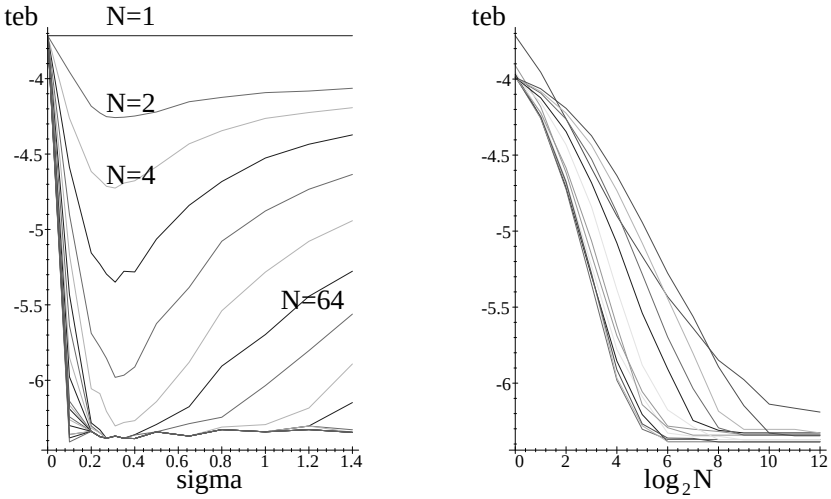
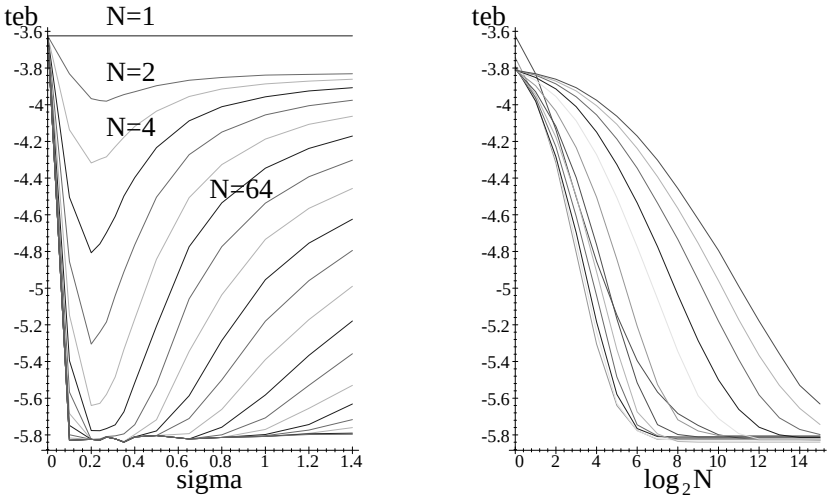


FIG. 3.6: $n = 180, R = 3/4, E_b/N_0 = 4\,\mathrm{dB}$



log_{10} teb en fonction de σ pour (de haut en bas) N =1,2,4,8...

log_{10} teb en fonction de $\log_2 N$ pour les différentes valeurs de σ

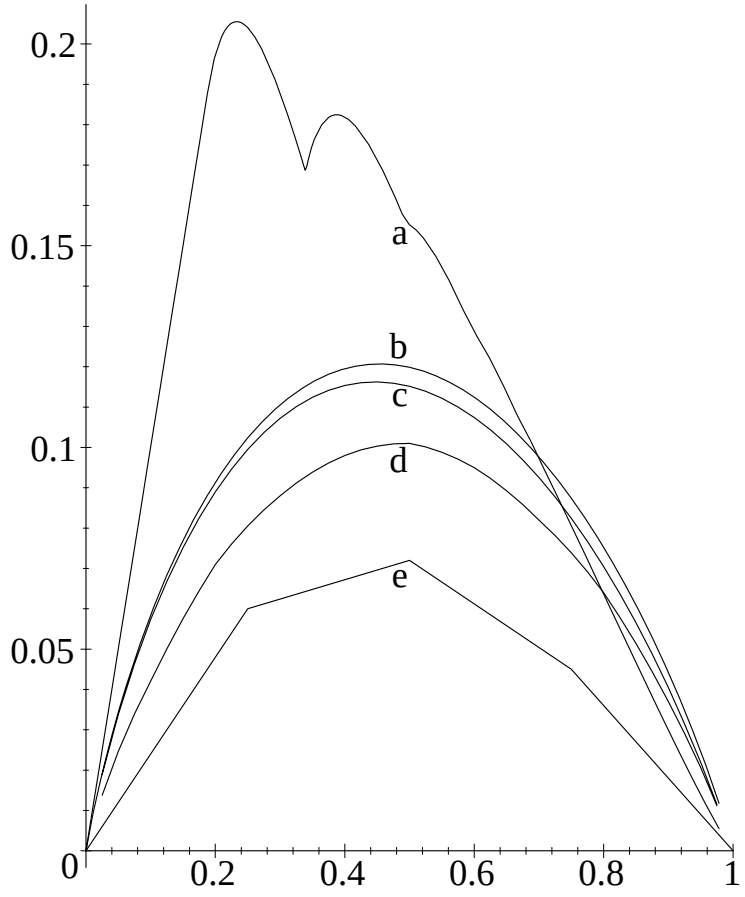


FIG. 3.7: *Coefficients de complexité: (a) Décodage par treillis; (b) décodage par ensemble d'information, (c) idem avec utilisation de codes poinçonnés, (d) idem avec utilisation de surcodes, (e) idem avec résonnance stochastique.*

Conclusion, perspectives

L'objectif annoncé était une utilisation optimale du canal, mais nous avons considéré que l'alphabet d'entrée du canal était binaire, avec des symbôles équiprobables, ou éventuellement quaternaire, les modulation BPSK ou QPSK étant équivalentes. Or Shannon a montré que, sous une contrainte de puissance moyenne à ne pas dépasser, l'alphabet d'entrée optimum pour le canal gaussien était continu avec une densité de probabilité normale (gaussienne) pour les différents symbôles.

Pour de faibles rapports signal à bruit, la différence est ténue, mais au delà, ne faudrait-il pas étudier des combinaisons de codes correcteurs d'erreur avec des codages «binaire à signal» alternatifs, ressemblant davantage à cet alphabet optimal ?

De notre étude sur le décodage dur, il semblait ressortir une mesure de «l'exponentialité» du problème du décodage. Cette mesure a-t-elle un sens ? Peut-on définir des classes de langages, telles NP ou $PCP(r(x), q(x))$ (cf. [Sud93]), ou de problèmes d'optimisation, par un coefficient de complexité minimum, en faisant au besoin l'hypothèse $P \neq NP$? Cela pourrait-il faire avancer les sciences de l'informatique ?

Cette étude a fait ressortir ces questions, mais son utilité était surtout d'apporter une bonne visibilité pour la définition d'algorithmes de décodage souple quasi-optimaux et généraux, afin de ne pas se perdre en considérations erratiques.

Nous avons en effet pu voir que le décodage souple, du point de vue transmission d'information, est nettement plus intéressant que le décodage dur, bien que moins abondamment traité dans la littérature. Le troisième chapitre doit donc être considéré comme plus important.

Il en est ressorti deux algorithmes peu ou pas connus de la communauté des codeurs nettement plus rapides que leur concurrents du décodage dur ou souple. Ont-ils un intérêt face aux turbo-codes ? Certainement.

En effet, les turbo-décodeurs de codes produits s'appuient sur des décodeurs souples de leurs codes internes. On voit aujourd'hui, implémentés expérimantalement ou utilisés en transmissions réelles, des codes produits dont les turbo-décodeurs utilisent par exemple les algorithmes de décodage de Fossorier et Lin. Ceux que nous avons vus permettraient de décoder plus rapidement ou d'utiliser des codes internes aléatoires de plus grande longueur.

Par ailleurs, les turbo-codes sont très satisfaisants lorsque l'on peut supporter

les délais que leur grande longueur impose, mais les transmissions par paquets qui sont et seront de plus en plus utilisées avec les nouvelles technologies de l'information requièrent des longueurs de bloc de quelques centaines en général et d'au plus quelques milliers. S'il était envisageable de décoder en temps réel de manière quasi-optimale des codes aléatoires de cette longueur, les résultats obtenus surpasseraient ceux des turbo-codes.

Or, tenant compte de la loi de Moore et du coefficient de complexité inférieur à 7% que nous avons obtenu pour les codes de taux 1/2, la longueur décodable maximale augmentera de plus de $1/0.07 \approx 14$ unités touts les dix-huit mois, disons d'une unité par mois, ou d'unité et demi pour des codes de taux 3/4.

Avec le décodage par résonnance stochastique, il est déjà possible, avec un simple programme en C, de décoder en temps réel un code de longueur 400 et de taux 1/2, ou de longueur 700 et de taux 3/4; les applications, avec de telles longueurs, ne manquent pas.

Pour terminer, j'aimerais soulever ce dernier point: le bruit secondaire, ajouté par l'algorithme de décodage par résonnance stochastique est un bruit blanc, gaussien, additif. Ce choix n'a été motivé par aucune raison particulière, il s'avère simplement que cela marche bien. Quel est le bruit secondaire optimal? Dans quel mesure dépend-il du canal?

Les démodulateurs utilisés pour la réception des signaux, ont pour la plupart une charge d'apprentissage du canal qui leur permet de mieux fonctionner, et peuvent parfois s'adapter à un canal qui varie dans le temps. Les problèmes posés par cet apprentissage ont été largement étudiés (voir par exemple [BMP87]) et assez bien résolus... Les décodeurs ne devraient-ils pas eux aussi être doués d'apprentissage? Et ne devraient-ils pas maintenant travailler plus en collaboration avec les démodulateurs (puisqu'après décodage, ils connaissent la forme du bruit, ils peuvent renseigner le démodulateur à ce propos, améliorant son apprentissage. Le démodulateur, quant à lui, fournit l'information souple au décodeur...)?

Deuxième partie

Détection et reconnaissance

Introduction

Je tiens ce monde pour ce qu'il est [...]: un théâtre où chacun doit jouer son rôle.
Shakespeare (Le marchand de Venise)

Derrière le monde dans lequel nous vivons, loin à l'arrière-plan, se trouve un autre monde ; leur rapport réciproque ressemble à celui qui existe entre les deux scènes qu'on voit parfois au théâtre, l'une derrière l'autre.
Søren Kierkegaard (Le Journal du séducteur)

Il ne fait aucun doute qu'il existe un monde invisible. Cependant, il est permis de se demander à quelle distance il se trouve du centre-ville et jusqu'à quelle heure il est ouvert.
Woody Allen (Dieu, Shakespeare... et moi)

Cette partie, consacrée à la détection et à la reconnaissance de code, a été écrite de manière à pouvoir être lue indépendamment de la partie traitant du décodage, à condition toutefois d'avoir une certaine connaissance des codes linéaires binaires, et à l'exception de certaines sections s'y reportant. Elle traite du problème suivant:

Etant donné un train binaire transmis à travers un canal dont on connaît le comportement statistique, et étant donnée une observation d'une partie connexe de ce train après traversée du canal, essayer de répondre aux questions suivantes:

Le train binaire etait-il, avant traversée du canal, la concaténation de mots binaires appartenant à un même code linéaire binaire ?

Et en cas de réponse positive, quelle est la longueur de ce code, quelle est la synchronisation des mots de code dans le train binaire, de quel code s'agit-il ?

L'intérêt de savoir y répondre se ressent lorsque l'on désire obtenir des informations à partir d'un train binaire intercepté sans connaître le protocole de communication (contexte d'espionnage).

Nous allons voir, après avoir formulé plus rigoureusement le problème de décision à résoudre, qu'il s'agit d'un problème NP-complet.

Le signal intercepté sera stocké en mémoire selon les capacités du démodu-

lateur soit sous la forme d'un train binaire démodulé soit sous une forme offrant une information souple, c'est-à-dire donnant pour chaque bit non pas sa valeur présumée mais sa probabilité de valoir 0 ou 1.

Si l'on sait, en faisant la bonne hypothèse de longueur et de synchronisation, se rendre compte du fait qu'un code linéaire binaire a bien été utilisé, on parviendra, en essayant toutes les hypothèses quant à la longueur et la synchronisation à effectuer simultanément la détection de code et la reconnaissance de longueur et de synchronisation. La reconnaissance de code, quant à elle, sera abordée sous l'angle d'une reconstruction des relations de parité.

Nous essaierons d'apporter une solution acceptable en surveillant de près les facteurs sensibles suivants: La complexité de l'algorithme, les probabilités de fausse alarme et de détection (pour la détection) et la probabilité de mauvaise reconnaissance. L'information souple sera très bénéfique et son traitement n'augmentera la complexité que par une constante.

Chapitre 1

Histoires de rang

J'écrirai ici mes pensées sans ordre, et non pas peut-être dans une confusion sans dessein: c'est le véritable ordre, et qui marquera toujours mon objet par le désordre même. Je ferais trop d'honneur à mon sujet, si je le traitais avec ordre, puisque je veux montrer qu'il en est incapable.

Blaise Pascal (section VI)

J'écrirai mes pensées avec ordre, par un dessein sans confusion. Si elles sont justes, la première venue sera la conséquence des autres. C'est le véritable ordre. Il marque mon objet par le désordre calligraphique. Je ferais trop de déshonneur à mon sujet, si je ne le traitais pas avec ordre. Je veux montrer qu'il en est capable. *Isidore Ducasse (Poésies II)*

Introduction

Un code linéaire binaire est un ensemble de mots (vecteurs) binaires, dits mots de code, de même longueur n, consituant un sous-espace-vectoriel strict de $\mathbf{F}_2^n$, l'ensemble, dit « espace ambiant », de tous les n⇔uplets binaires.

Pour étudier une hypothèse de longueur n et de synchronisation donnée, nous extrairons une $n \times N$⇔matrice X du train binaire en y découpant des mots de longueur n selon cette hypothèse.

Pour chaque hypothèse de longueur et de synchronisation, l'hypothèse, que nous noterons $\mathcal{H}$, à valider est que les lignes de X étaient éléments d'un même code linéaire binaire de longueur n (c'est à dire d'un sous espace vectoriel strict de l'espace ambiant) avant de traverser le canal. Bien sûr, pour que cela ait un sens, il faut qu'au moins n mots aient été émis (sinon l'hypothèse ainsi énoncée est toujours vraie). Nous supposerons donc que $N > n$.

Un critère de détection incontournable sera donc que les lignes de X soient elles-mêmes éléments d'un même sous espace vectoriel strict de l'espace ambiant, c'est à dire que la matrice X soit de rang non plein.

Nous allons étudier ce critère pour nous rendre compte qu'en pratique, il sera insuffisant. Puis nous ferons quelques remarques sur le problème posé par le fait que la longueur, la synchronisation et la dimension de l'hypothétique code sont inconnues, et sur ce que l'on peut en dire *a priori*. Enfin nous montrerons que le problème à résoudre, même si l'on connaît la longueur, la synchronisation et la dimension de l'hypothétique code, est NP-complet.

1.1 Critère du rang

L'algorithme de détection le plus simple consiste à décider que l'on est en présence d'un codage linéaire si la $N \times n$⇔matrice X ($N > n$) est de rang non plein, c'est à dire de rang strictement plus petit que n.

Nous sommes alors confrontés à deux sources d'erreur: d'une part, le critère pourrait être vérifié même si l'hypothèse est fausse; d'autre part, même si l'hypothèse est juste, les erreurs de transmission peuvent être telles que la matrice X soit en fait de rang plein.

Nous allons étudier les probabilités, dites respectivement de fausse alarme et de non-détection, associées à ces deux évènements.

1.1.1 Probabilité de fausse alarme

La probabilité de fausse alarme, notée P_{fa}, est la probabilité qu'une $N \times n$⇔matrice binaire quelconque soit de rang non plein. On trouvera le dénombrement des matrices q-aires de dimensions et de rangs donnés par exemple dans [LN83] (p. 455), et ainsi vérifier la proposition suivante.

Proposition 10

$$P_{fa} = \Pr(\mathrm{rk}(\Lambda_N) < n) = 1 \Leftrightarrow \prod_{i=0}^{n-1} (1 \Leftrightarrow 2^{i-N})$$

Corollaire 1

$$2^{n-N} \Leftrightarrow 2^{-N} \Leftrightarrow 2^{2(n-N)} < P_{fa} < 2^{n-N} \Leftrightarrow 2^{-N}$$

Démonstration: (du corollaire) On a:

$$1 \Leftrightarrow \sum_{i=0}^{n-1} 2^{i-N} < \prod_{i=0}^{n-1} (1 \Leftrightarrow 2^{i-N}) < 1 \Leftrightarrow \sum_{i=0}^{n-1} \left(2^{i-N} \Leftrightarrow \sum_{j=i+1}^{n-1} 2^{i-N} 2^{j-N} \right)$$

Donc $\sum_{i=0}^{n-1} \left(2^{i-N} \Leftrightarrow \sum_{j=i+1}^{n-1} 2^{i-N} 2^{j-N} \right) < P_{fa} < \sum_{i=0}^{n-1} 2^{i-N}$.

Or $\sum_{i=0}^{n-1} 2^{i-N} = 2^{n-N} \Leftrightarrow 2^{-N}$, et $\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 2^{i-N} 2^{j-N} < 2^{2(n-N)}$. On en déduit l'inégalité du corollaire. □

Ainsi, pour avoir une probabilité de fausse alarme inférieure par exemple à $1/1000$, il faut et il suffit que N soit au moins égal à $n+10$.

1.1.2 Probabilité de détection

Tout d'abord, nous démontrons un lemme qui nous sera utile à maintes reprises dans l'ensemble de ce document du fait que nous travaillons sur le corps binaire. Il est effectivement lié à la probabilité pour qu'une variable aléatoire prenne un nombre pair de fois une certaine valeur parmi un nombre donné de réalisations, si bien qu'on y revient souvent lorsque l'on traite de probabilités en rapport avec les relations de parité (cf. par exemple le lemme 1 de l'article original de Gallager sur les codes à relations de parité de faible densité [Gal62]).

Lemme 3

$$\forall n \in \mathbf{N} \qquad \forall x \in \mathbf{R} \qquad \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{2i} x^{2i} (1 \Leftrightarrow x)^{n-2i} = \frac{1 + (1 \Leftrightarrow 2x)^n}{2}$$

Démonstration:

$$\begin{aligned}
\text{On a} \quad (1 \Leftrightarrow 2x)^n = ((1 \Leftrightarrow x) \Leftrightarrow x)^n = & \sum_{i=0}^{n} \binom{n}{i} (\Leftrightarrow 1)^i x^i (1 \Leftrightarrow x)^{n-i} \\
\text{et} \qquad 1 = ((1 \Leftrightarrow x) + x)^n = & \sum_{i=0}^{n} \binom{n}{i} x^i (1 \Leftrightarrow x)^{n-i} \\
\text{donc} \qquad \frac{1+(1-2x)^n}{2} = & \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{2i} x^{2i} (1 \Leftrightarrow x)^{n-2i}.
\end{aligned}$$

□

Ainsi, si le canal présente un taux d'erreur binaire, ou probabilité de transition égal à τ, la probabilité pour que le nombre de positions erronées parmi w positions données soit pair vaut $\frac{1+(1-2\tau)^w}{2}$. Le terme $(1 \Leftrightarrow 2\tau)$ étant beaucoup utilisé dans la suite, nous définissons $z = 1 \Leftrightarrow 2\tau$.

Appelons $\mathcal{H}$ l'hypothèse à détecter (selon laquelle le train binaire etait, avant traversée du canal, la concaténation de mots binaires appartenant à un même code linéaire binaire). La probabilité de détection, notée P_{det}. est la probabilité, dans le cas où $\mathcal{H}$ est vraie, pour que la matrice X soit de rang non plein.

Supposons $\mathcal{H}$ vraie et appelons C le code qui a été utilisé.

Proposition 11 *Soit, pour $1 \leq \omega \leq n$, $C_\omega^\perp$ le nombre de mots de poids ω dans $C^\perp$, le dual de C. On a:*

$$P_{det} < \sum_{\omega=1}^{n} C_\omega^\perp \left(\frac{1+z^w}{2} \right)^N + 2^{n-N} \Leftrightarrow 2^{n-k-N}$$

Démonstration: X est de rang non plein si et seulement si il existe un mot h non nul tel que $hX^t = 0$. On a (borne de l'union):

$$P_{det} = \Pr(\exists h \in \mathbf{F}_2^\star / h\Lambda_N^t = 0 | \mathcal{H}) < \sum_{h \in \mathbf{F}_2^\star} \Pr(h\Lambda_N^t = 0 | \mathcal{H})$$

Or, pour tout $h \in C^\perp$ de poids ω, $\Pr(h\Lambda_N^t = 0)$ est la probabilité pour que les N motifs d'erreur soient tous de poids pair sur le support de h, c'est à dire $\left(\frac{1+z^w}{2}\right)^N$.

Et pour tout h non nul et non élément de $C^\perp$, $\Pr(h\Lambda_N^t = 0) = 2^{-N}$. □

Cette probabilité de détection sera vite trop faible pour les taux d'erreur binaires (τ) usuels, c'est à dire que le critère du rang est trop strict.

1.2 Oracle du rang

1.2.1 Remarques sur la longueur et la synchronisation

Il sera souvent possible de connaître *a priori* la longueur et la synchronisation des hypothétiques mots de code.

Tel peut être le cas si par exemple une mise en trame du train binaire est détectée. La détection de mise en trame n'est pas abordée dans cette thèse mais il faut savoir que la plupart des protocoles de transmission comprennent de nombreuses sophistications quant au format des trains binaires devant transiter dans le canal, dont une utilité (parmi d'autres) est de permettre de récupérer la synchronisation (le train binaire contient même assez souvent en préfixe des informations sur sa nature et sur son encodage).

Supposons que ce ne soit pas le cas et que le train binaire émis (dépourvu d'erreur) $M = (m_1 \dots m_l)$ soit simplement une concaténation de mots de code, et appelons $M_{n,s}$ ($0 \leq s < n$, $s + n^2 < l$) la matrice extraite de M selon l'hypothèse de longueur n et de synchronisation s et $N_{n,s} = \lfloor \frac{l-s}{n} \rfloor$ le nombre de lignes correspondant:

$$M_{n,s} = \begin{pmatrix} m_{s+1} & m_{s+2} & \dots & m_{s+n} \\ m_{s+n+1} & m_{s+n+2} & \dots & m_{s+2n} \\ \dots & \dots & \dots & \dots \\ m_{s+(N_{n,s}-1)n+1} & \dots & \dots & m_{s+N_{n,s}n} \end{pmatrix}$$

Si C est le code qui a été utilisé, et si (n, s) est la bonne hypothèse de longueur et de synchronisation alors $\mathrm{rk}(M_{n,s}) \leq \dim(C) < n$.

Supposons que l'on dispose d'un oracle capable de détecter cette non-plénitude du rang, même aprés addition d'une erreur de transmission (dans la suite, nous allons en quelque sorte nous efforcer de construire un algorithme de détection se rapprochant de cet oracle), palliant ainsi à la faible probabilité de détection du critère de la section précédente. A quelles nouvelles difficultés sommes-nous alors confrontés?

Que se passe-t-il si l'on fait une mauvaise hypothèse de longueur ou de synchronisation ?

Une mauvaise hypothèse de longueur peut faire perdre toute structure à la matrice extraite. Il existe des cas où la longueur réelle n, la dimension k et la longueur testée n' sont telles que la matrice extraite (que l'on suppose ici dépourvue d'erreur) puisse, avec une probabilité non négligeable, être de rang non plein; par exemple si $\left(1 + \lceil \frac{n'-1}{n} \rceil\right) k < \mathrm{pgcd}(n, n')$.

Ces cas sont cependant fort marginaux, et il est de toute façon quasiment (les algorithmes seront non déterministes, rien n'est exclu à 100 %) impossible que l'évaluation de la mauvaise hypothèse supplante et dissimule celle de la bonne; encore faut-il mener cette dernière à bien, c'est pourquoi on testera exhaustivement les hypothèses de longueur même si, en cours de calcul, l'une d'entre elles se révèle crédible.

Une erreur sur la synchronisation serait en revanche moins dramatique. On a en effet pour $0 \leq s' < n$: $\mathrm{rk}(M_{n,s'}) \leq \mathrm{rk}(M_{n,s}) + |s \Leftrightarrow s'| \leq \dim(C) + |s \Leftrightarrow s'|$ (ce que l'on montre facilement en considérant que $\mathrm{rk}([A|B]) \leq \mathrm{rk}([A]) + \mathrm{rk}([B])$). Si l'erreur de synchronisation n'est pas trop grande (ni la dimension du code), la détection sera donc toujours possible.

Il est par ailleurs possible d'éviter d'avoir à tester de nombreuses hypothèses de synchronisation en considérant les matrices M'_n à $2n \Leftrightarrow 1$ colonnes définies de la manière suivante:

$$M'_n = \begin{pmatrix} m_1 & m_2 & \cdots & m_{2n-1} \\ m_{n+1} & m_{n+2} & \cdots & m_{3n-1} \\ \cdots & \cdots & \cdots & \cdots \\ m_{(N_{n,n-1}-1)n+1} & \cdots & \cdots & m_{N_{n,n-1}n+n-1} \end{pmatrix}$$

Elle contient en effet la matrice $M_{n,s}$ qui nous intéresse qui est de rang au plus $\dim(C)$. La matrice complémentaire à $n \Leftrightarrow 1$ colonnes étant de rang au plus $n \Leftrightarrow 1$, on a donc $\mathrm{rk}(M'_n) \leq \dim(C) + n \Leftrightarrow 1 < 2n \Leftrightarrow 1$ et notre oracle pourrait nous indiquer que l'hypothèse de longueur est bonne. Il ne resterait plus alors qu'à retrouver la synchronisation.

Si aucune mise en trame n'est détectée auparavant sur le train binaire, et que la détection doit s'opérer « à l'aveugle », c.a.d. sans connaissance de la longueur et de la synchronisation, alors la meilleure méthode de détection que je puisse envisager est une détection accompagnée d'une reconnaissance de longueur et de synchronisation.

1.2.2 Remarques sur la dimension

Pour définir des critères pertinents, nous allons faire une étude de certaines probablités. Notre but ne sera pas d'accomplir une étude parfaite et des calculs exacts, mais d'avoir une idée de ce qui est négligeable et de l'ordre de grandeur de ce qui ne l'est pas, afin d'avoir un critère sous la forme d'une fonction à faire calculer à l'algorithme.

Ces ordres de grandeur dépendront pour certains dénombrements de la dimension k du code à détecter, que l'on ne connaît pas. Pour pallier à ce défaut

lorsqu'il se présentera, nous aurons recours une fois de plus à un calcul de probabilité, mais nous perdrons grandement en précision, les dépendances en la dimension étant généralement exponentielles.

Nous supposons connu le taux d'erreur binaire τ, ou le rapport signal à bruit $\rho = \mathcal{E}^2/N_0$ ($\mathcal{E}^2$ est l'énergie par bit transmis et non l'énergie par bit d'information qui est généralement considérée dans la littérature sur le codage de canal). Supposons en outre que ce paramètre soit le même que pour le destinataire, ou du moins que l'on connaisse le rapport signal transmis à bruit ρ_d dont celui-ci dispose (ce rapport est généralement variable, le « pire cas » est alors considéré). Si l'on minore la qualité de transmission dont il est supposé bénéficier, si l'on suppose par exemple qu'un taux d'erreur résiduel inférieur à 10^{-3} est attendu, on peut alors majorer le taux de codage, et si l'on connaît la longueur du code on pourra majorer de manière plus fine sa dimension:

En l'absence d'hypothèse sur la longueur, tout ce que l'on sait est que, si R désigne le taux de codage, la quantité $10\log_{10}(\rho_d) \Leftrightarrow 10\log_{10}(R) > L_{Sh}(R) + 1.5$dB, où $L_{Sh}(R)$ est la limite de Shannon (cf. partie «Décodage», section 1.1.2, p. 23). En effet, transmettre à moins de 1.5 dB de cette limite en utilisant un code linèaire binaire de longueur inférieure à 1000, ne permettrait pas d'atteindre un taux d'erreur résiduel inférieur à 10^{-3}, même en utilisant le meilleur décodage qui soit pour les codes linéaires binaires, à savoir le décodage souple à maximum de vraisemblance. On en déduit que $10\log_{10}(R) + L_{Sh}(R) < 10\log_{10}(\rho_d) \Leftrightarrow 1.5$, ce qui nous donne une majoration $R_{max}(\rho_d)$ du taux de codage.

Quant aux codes linéaires binaires de longueur supérieure à 1000, nous ne parviendrions de toute façon pas à les détecter pour des causes de temps de calculs et de longueur de train à intercepter.

Si nous avons une hypothèse sur la longueur n, bien-sûr nous savons en vertu de ce qui précède que la dimension est inférieure $R_{max}(\rho_d)n$, mais nous pouvons être plus précis en évaluant pour chaque k le meilleur taux d'erreur que l'on pourrait atteindre avec le triplet (n, k, ρ_d), et en supposant qu'il doit être inférieur à 10^{-3}.

Ce taux d'erreur est celui qui serait atteint par un algorithme de décodage souple à maximum de vraisemblance. L'estimation la plus précise de ce dernier est fournie par la borne tangentielle sphérique [Pol94]. Le lecteur pourra se reporter à la section 1.1.4, p. 30, de la partie «Décodage» de ce manuscrit pour plus de précisions.

Nous pouvons ainsi calculer $k_{max}(\rho_d, n)$ la valeur maximale que puisse prendre la dimension du code. Par ailleurs, il est dans l'intérêt des protagonistes de la transmission (pour une question de rendement) d'utiliser la plus grande dimension possible (qui est strictement inférieur à $k_{max}(\rho_d, n)$ si leur algorithme de décodage est moins performant que le décodage souple à maximum de vraisemblance ou s'ils attendent un taux d'erreur strictement inférieur à 10^{-3} ou si la famille de code utilisée n'offre pas la possibilité de choisir une dimension égale à $k_{max}(\rho_d, n)$).

Nous pouvons aussi éventuellement calculer $k_{opt}(\rho_d, n) \leq k_{max}(\rho_d, n)$ qui est la valeur la plus probable de la dimension sachant que tel taux d'erreur est attendu et que tel algorithme de décodage est utilisé, ou telle famille de codes...

Si l'on appelle D_n l'hypothèse selon laquelle la longueur du code recherché est n et $D_{k,n}$ l'hypothèse selon laquelle la dimension du code vaut k et sa longueur n, on définira une distribution de probabilité pertinente pour cette dimension en utilisant les quelques relations suivantes:

$$\forall k > k_{max}(\rho_d, n) \quad \Pr(D_{k,n}|\,\mathcal{H}) = 0,$$

$$\sum_{k=1}^{k_{max}(\rho_d,n)} \Pr(D_{k,n}|\,\mathcal{H}) = \Pr(D_n|\,\mathcal{H}),$$

$$k_1 < k_2 \leq k_{opt}(\rho_d, n) \Longrightarrow \Pr(D_{k_1,n}|\,\mathcal{H}) < \Pr(D_{k_2,n}|\,\mathcal{H}).$$

On peut raffiner cette étude en associant à tout couple (k, n) les trois valeurs suivantes: $R(k, n) = \frac{k}{n}$ le taux de codage, $P_{ML}(k, n)$ le taux d'erreur après décodage à maximum de vraisemblance, et , (k, n) un minorant pour la complexité du décodage (par exemple n^2).

Un code sera plus vraisemblablement utilisé qu'un autre s'il est plus économique ou plus performant. La fonction $\Pr(D_{k,n}|\,\mathcal{H})$ est donc croissante en $R(k, n)$ et décroissante en $P_{ML}(k, n)$ et , (k, n). On pourra prendre:

$$\Pr(D_{k,n}|\,\mathcal{H}) = Kf(k/n)g(P_{ML}(k, n))h(,\,(k, n))$$

où $f(R)$ est une fonction croissante, par exemple $f(R) = R$, $g(P)$ est décroissante, par exemple $g(P) = 1$ si $P < 10^{-3}$ et 0 sinon, $h(,\,)$ est décroissante, par exemple $h(,\,) = ,^{-x}$ $(x > 0)$, et la constante K est telle que $\sum_{k,n} \Pr(D_{k,n}|\,\mathcal{H}) = 1$.

On définira alors $k_{\max} = \max k/\Pr(D_{k,n}|\,\mathcal{H}) > 0$ et $k_{\min} = \min(k/\Pr(D_{k,n}|\,\mathcal{H}) > 1/n)$ par exemple.

Je laisserai pour le moment intact ces degrés de liberté, car toute considération plus avant sur la dimension ou la longueur porterait à caution ou dépend du contexte précis de la détection.

1.3 Réduction de rang, NP-complétude du problème

Nous avons vu que le critère du rang était trop sévère. Pour avoir une probabilité de détection suffisante, il faut pouvoir tolérer, d'une manière ou d'une autre, les erreurs de transmission. Par ailleurs, pour que cette tolérance n'augmente pas démesurément la probabilité de fausse alarme, il faudra supposer, non pas seulement que la matrice était de rang non plein avant transmission, mais qu'elle était de rang au plus k_{max}, pour $k_{max} < n$ supposé connu.

On pourra alors décider que l'hypothèse est juste s'il existe un sous-espace vectoriel de dimension k_{max} tel que la somme des distances des mots reçus à ce sous-espace vectoriel est inférieur à un certain seuil. Nous étudions ici spécifiquement ce problème que nous appelons « Réduction de rang ».

1.3.1 Notations et définition du problème

Nous noterons $\mathrm{rk}(M)$ le rang d'une matrice M, et nous appellerons poids de cette matrice et nous noterons $\mathrm{wt}(M)$ le nombre de ses coefficients non nuls.

Soient N et n deux entiers positifs, et X une $N \times n$-matrice binaire. Soit $k \leq \mathrm{rk}(X)$ un entier positif. Le problème de réduction de rang, RR, peut s'énoncer ainsi:

RR(X,k) *Trouver une $N \times n$-matrice binaire $E^\star$ de poids minimum, et vérifiant:* $\mathrm{rk}(X+E^\star) \leq k$.

Si w est un entier, le problème de décision associé à RR est:

DRR(X,k,ω) *Existe-t-il une $N \times n$-matrice binaire $E^\star$ vérifiant:*

$$\begin{cases} \mathrm{rk}(X+E^\star) \leq k \\ \mathrm{wt}(E^\star) \leq \omega \end{cases}$$

1.3.2 Complexité

DRR est clairement dans la classe NP puisque, si la réponse à $\mathrm{DRR}(X,k,\omega)$ est positive, étant donné une solution $E^\star$ (comme certificat), on peut vérifier en temps polynomial que $\mathrm{rk}(X+E^\star) \leq k$ et que $\mathrm{wt}(E^\star) \leq \omega$.

Nous allons prouver que DRR est NP-complet en utilisant la NP-complétude du problème de décision DDM associé à la distance minimale d'un code [Var97].

Soient n et k deux entiers positifs, $k < n$, G une $k \times n$-matrice binaire et soit ω un entier.

DDM(G,ω) *Existe-t-il $c \in \mathrm{span}(G)$, non nul, vérifiant* $\mathrm{wt}(c) \leq \omega$.

où $\mathrm{span}(G)$ *est le s.e.v engendré par les lignes de G.*

Proposition 12 *Soit n et k deux entiers, $k < n$. Soit G une $k \times n$-matrice binaire vérifiant* $\mathrm{rk}(G) = k$. *On a alors: DDM(G,ω)=DRR$(G, k \Leftrightarrow 1, \omega)$.*

Démonstration: Supposons que la réponse à $\mathrm{DRR}(G, k \Leftrightarrow 1, \omega)$ soit positive.

Soit alors $(G_1 \cdots G_k)$ les lignes de G, et $E^\star$ une matrice telle que:

$\begin{cases} \mathrm{rk}(G+E^\star) \leq k \Leftrightarrow 1 \\ \mathrm{wt}(E^\star) \leq \omega \end{cases}$ et $(E_1^\star, \cdots, E_{k+1}^\star)$ ses vecteurs lignes.

Puisque $\mathrm{rk}(G+E^\star) < \mathrm{rk}(G)$, il existe $(\lambda_1, \cdots, \lambda_k) \in F_2^k$ tels que:

$$\begin{cases} \sum_{i=1}^k \lambda_i (G_i + E_i^\star) = 0 \\ \sum_{i=1}^k \lambda_i G_i \neq 0 \end{cases}$$

donc $\sum_{i=1}^k \lambda_i E_i^\star \left(= \sum_{i=1}^k \lambda_i G_i \right)$ est un vecteur non nul de $\mathrm{span}(G)$.

or $\mathrm{wt}(\sum_{i=1}^{k} \lambda_i E_i^\star) \leq \mathrm{wt}(E^\star) \leq \omega$, donc la réponse à $\mathrm{DDM}(G, \omega)$ est positive.

Si au contraire la réponse à $\mathrm{DRR}(G, k \Leftrightarrow 1, \omega)$ est négative, nous savons que:

$$\forall E \qquad \mathrm{rk}(G+E) \leq k \Leftrightarrow 1 \Rightarrow \mathrm{wt}(E) > \omega$$

Soit alors $c = \sum_{i=1}^{k} \lambda_i G_i$ un vecteur non nul de $\mathrm{span}(G)$, prouvons que $\mathrm{wt}(c) > \omega$:

Soit $p \in \{1...k\}$ tel que $\lambda_p \neq 0$ et soit E la $k \times n$-matrice dont la p ième ligne est égale à c et dont les autres lignes sont nulles. On a $\sum_{i=1}^{k} \lambda_i (G_i + E_i) = c + \sum_{i=1}^{k} \lambda_i G_i = 0$, donc $\mathrm{rk}(G+E) < k$ et par conséquent $\mathrm{wt}(c) = \mathrm{wt}(E) > \omega$.

La réponse à $\mathrm{DDM}(G, \omega)$ est donc négative. □

Théorème 10 *DRR est NP-complet.*

Démonstration: Nous avons vu que DDM peut se réduire à DRR si la matrice est de rang plein. Supposons que nous ayons un algorithme résolvant DRR en temps polynomial et soient (G, w) une instance du problème DDM, et G' une matrice de rang plein telle que $\mathrm{span}(G') = \mathrm{span}(G)$. Alors, en résolvant, grâce à notre algorithme, le problème $\mathrm{DRR}(G', \mathrm{rk}(G) \Leftrightarrow 1, w)$, nous sommes en mesure de répondre en temps polynomial au problème $\mathrm{DDM}(G, w)$.

Un algorithme en temps polynomial résolvant DRR induit donc un algorithme en temps polynomial résolvant DDM (de même donc, puisque DDM est NP-complet, pour tout problème de la classe NP).

Ceci prouve (avec le fait que DRR est dans la classe NP) que DRR est NP-complet. □

Conclusion

La première section a introduit les concepts de probabilité de fausse alarme et de détection dont nous allons nous servir abondamment par la suite. Ils nous ont permis de mesurer «l'insuffisance» du critère du rang, et seront utilisés à la même fin pour tout autre critère.

Nous avons vu ensuite une forme pratique à donner à la matrice X, qui premttra, lors de la détection, de ne pas se poser la question de la synchronisation. Cependant, nous oublierons ce détail dans la suite pour rendre plus compréhensible nos propos.

Nous avons également défini $k_{\min}$ et $k_{\max}$ les valeurs minimales et maximales de la dimension du code à détecter, s'il en est, ainsi que les probabilités $\Pr(D_{k,n} | \mathcal{H})$ pour que la longueur et la dimension de l'hypothétique code soient respectivement n et k et $\Pr(D_n | \mathcal{H}) = \sum_{k=1}^{n-1} \Pr(D_{k,n} | \mathcal{H})$ la probabilité pour quela longueur soit n. Nous y ferons plusieurs fois référence.

Enfin, nous avons un nouveau venu dans la classe des problèmes NP-complets, le problème de la réduction de rang, qui est directement lié à notre problème. Nous ne pourrons donc résoudre que les instances «faciles» de celui-ci, c.a.d. qu'il sera nécessaire que le taux d'erreur binaire ne soit pas trop élevé ou que la portion de train binaire considérée soit «propre».

Chapitre 2

Détection et reconnaissance: généralités

[...] il est de certaines sensations délicieuses dont le vague n'exclut pas l'intensité; et il n'est pas de pointe plus acérée que celle de l'Infini.
Charles Baudelaire

Rien de noble ne se fait sans hasard.
Michel de Montaigne (Essais 1.24)

J'entendrai des regards que vous croirez muets.
Jean Racine

Introduction: inconnu et hasard

Comme c'est souvent le cas en théorie de l'information, les généralités qui seront traitées dans ce chapitre sont en rapport avec la notion de hasard. Les inconnues seront modélisées par des variables aléatoires dont les lois statistiques seront données.

Ainsi, on modélise une source d'information par des tirages successifs à pile ou face dont le nombre mesure la quantité d'information ainsi produite; de même on modélise un canal par un additionneur de bruit, le bruit étant lui-même un processus aléatoire obéissant à certaines lois statistiques.

Si le décodage consiste à déterminer les bits qui ont été «choisis par le hasard» de la source connaissant les statistiques du canal et les règles de codage, la reconnaissance de code consiste quant à elle à déterminer les règles de codage connaissant les statistiques du canal, de la source...

Pour ce problème les sources de hasard sont donc plus nombreuses qu'habituellement. Nous allons considérer, en plus de la source et du canal, deux autres sources de hasard. La première est l'émetteur: le train binaire intercepté est

d'origine inconnue, l'émetteur est un émetteur aléatoire parmi l'ensemble des émetteurs, les paramètres de la transmission sont donc eux-mêmes aléatoires, et en particulier le code (en tant qu'élément de l'ensemble de tous les codes, qu'objet combinatoire, qu'entité algébrique à part entière) qui a été utilisé, les mots de ce code (considérés individuellement, chacun pouvant être défini comme étant l'intersection de tous les codes qui le contiennent), et de même les mots de son code dual, c.a.d. les «relations de parité» qui doivent être satisfaites par la totalité des mots de code.

Une autre source de hasard que nous seront amenés à considérer est le pseudo-aléa qui sera utilisé par les algorithmes que nous mettrons au point et qui sera supposé être un vrai aléa.

Les fruits (ou réalisations) de ces sources de hasard peuvent être combinés entre eux, d'une manière imposée ou choisie, ils peuvent être observés sous un angle particulier, imposé ou choisi également. Si nous associons des variables aléatoires (VA) aux différentes sources de hasard, alors ces combinaisons ou ces angles d'observation sont en quelque sorte des opérateurs s'appliquant aux VA. Nous considèrerons leurs résultats comme des VA également, et nous aurons à notre disposition des réalisations de ces dernières.

Nous utiliserons alors le formalisme intuitif des probabilités: la probabilité pour qu'une VA prenne une valeur donnée correspondra à la fréquence attendue de cet évènement pour un nombre infini de réalisations identiques et indépendantes et en supposant exact le modèle statistique donné (pour les sources de hasard). Ce formalisme met à notre disposition l'axiomatique intuitive bien connue des probabilités que je ne rappellerai pas ici (je ne parle pas de la rigoureuse théorie axiomatique des probabilités mais des règles de calcul, de dénombrement, de prise en compte de conditions,... qui sont enseignées au lycée).

Nous verrons alors, en s'étant simplement donné ces quelques règles et en ayant défini la nature des problèmes de détection et de reconnaissance sans rentrer dans le détail de problèmes particuliers, qu'il est déjà possible de définir les principaux outils dont on se servira, et d'énoncer d'importants résultats.

Lorsqu'ensuite, nous considèrerons un problème particulier rentrant dans une des catégories de problèmes que nous allons définir dans ce chapitre, nous pourrons utiliser directement ces résultats. Nous parviendrons ainsi à donner la marche à suivre pour la détection et la reconnaissance de code.

2.1 Problèmes de détection

2.1.1 Définition

Considérons un phénomène $A : \emptyset \to Y$ (un algorithme, un canal... ou tout phénomène, déterministe ou non), générant spontanément un element de Y. Et considérons de même ce que nous appellerons l'évènement fortuit $\emptyset \to Y$, qui est en quelque sorte la réunion de tous les phénomènes générant un element de Y.

Supposons que l'on dispose d'un modèle probabiliste décrivant de manière

pertinente le phénomène A et l'évènement fortuit, c.a.d. permettant de calculer, pour tout élément y de Y, la probabilité pour qu'un élément de Y généré par la phénomène A soit égal à y et la probabilité, que nous noterons $\Pr(y)$, pour qu'un élément de Y généré de manière fortuite soit égal à y.

Etant donné un tel phénomène A, nous appellerons « algorithme de détection associé à A », un algorithme dont le but est de décider, observant un élément de Y, si le phénomène A est arrivé ou s'il s'agit d'un évènement fortuit.

Dans notre problème, le phénomène à détecter est l'utilisation d'un code linéaire binaire pour transmettre un message à travers un canal. L'ensemble Y est l'ensemble de tous les trains binaires. A consiste en la transmission d'un message m quelconque à travers un canal donné après encodage par un code linéaire binaire C quelconque et concaténation des mots de code obtenus.

Plus précisément, nous étudierons le cas d'un canal binaire symétrique de probabilité de transition τ connue (cas dur) et celui d'un canal à bruit blanc gaussien additif de rapport signal à bruit $\mathcal{E}^2/N0$ connu (cas souple); et nous supposerons que le code utilisé, s'il en est, l'a été à des fins de corrections d'erreur (et est donc adapté au canal).

2.1.2 Critère de détection, probabilités de détection et de fausse alarme

La réponse de l'algorithme de détection dépend uniquement de l'élément de Y qu'on lui donne à observer et qui aura donc à satisfaire un certain critère (dit « critère de détection ») pour que la réponse soit positive.

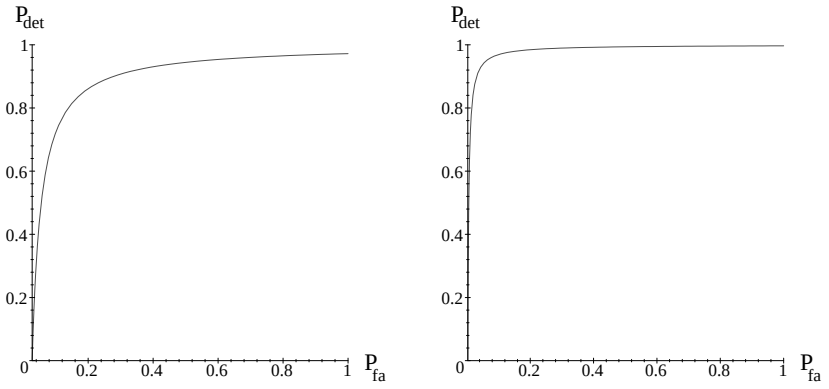
Le critère de détection est généralement basé sur un couple (F, S) avec F une fonction de Y dans $\mathbf{R}$ et S un réel appelé « seuil ». L'algorithme de détection, observant $y \in Y$, décidera que A a eu lieu si $F(y) \geq S$.

Les probabilité de détection et de fausse alarme sont des outils servant à évaluer l'efficacité d'un algorithme de détection.

Définition 11 *Etant donné un phénomène $A : \emptyset \rightarrow Y$ et un algorithme de détection associé à A, nous appellerons probabilité de fausse alarme et noterons P_{fa} la probabilité pour qu'un élément quelconque (fortuit) de Y satisfasse le critère de détection. De même, nous appellerons probabilité de détection et noterons P_{det} la probabilité pour qu'une sortie de A satisfasse ce même critère de détection.*

Bien sûr, l'algorithme de détection est efficace si la probabilité de fausse alarme est proche de zéro et la probabilité de détection proche de 1.

Ces probabilités varient simultanément lorsque l'on fait varier le seuil et sont donc généralement reliées monotoniquement entre elles. En effet, si l'on veut une très faible probablité de fausse alarme, on définira un seuil très haut, mais alors la probabilité de détection sera elle même faible. Inversement, si l'on veut une probabilité de détection très élevée, il faudra un seuil très bas, mais alors la probabilité de fausse alarme sera élevée.



(a) faible discriminance (b) haute discriminance

FIG. 2.1: *Probabilité de détection en fonction de la probabilité de fausse alarme*

En particulier, une probablité de fausse alarme égale à 0 correspond toujours à un seuil si élevé que la probablité de détection vaut elle-même 0 (l'algorithme répond négativement quelle que soit son entrée). De même, une probablité de détection égale à 1 correspond généralement (si l'image de A est Y tout entier, comme c'est souvent le cas lorsqu'un phénomène stochastique, tel un canal bruité, entre en jeu dans le phénomène à détecter) à une probabilité de fausse alarme elle-même égale à 1.

En réglant le seuil, on choisit donc un compromis entre ces deux probabilités. La figure 2.1 montre comment elles sont typiquement reliées entre elles. Ce genre de courbe relie toujours les points $(0,0)$ et $(1,1)$. La seconde courbe, plus « pointue » que la première, représente un critère de détection plus discriminant.

Pour un couple (P_{fa}, P_{det}) donné, $\frac{P_{det}}{P_{fa}}$ est la pente de la droite reliant l'origine au point de la courbe déterminé par ce couple. Bien entendu, ce rapport est en principe supérieur à 1.

Si l'on exprime P_{det} en fonction de $\frac{P_{det}}{P_{fa}}$, la fonction obtenue, de $[1, \infty[$ dans $]0,1]$, vaut 1 en 1, est décroissante, et tend vers 0. La décroissance est d'autant plus lente que le critère est discriminant (ce qui constitue une caractérisation de la notion de discriminance relative de deux critères mathématiquement plus correcte que la notion de courbe plus ou moins « pointue »).

2.1.3 Détection optimale

Essayons maintenant de voir ce que doit calculer la fonction F:

Appelons $\mathcal{H}$ l'hypothèse selon laquelle A a bien eu lieu et considérons l'élément observé comme une variable aléatoire: Pour tout $y \in Y$, on note $\Pr(y)$ la

probabilité d'observer y en dehors de toute hypothèse et $\Pr(y|\,\mathcal{H})$ la probabilité d'observer y si $\mathcal{H}$ est vraie (Dans le cas d'un ensemble Y non discret, on considère de manière similaire les densités de probabilité).

Soit Ω le sous-ensemble de Y contenant tous les éléments qui provoqueront une réponse positive de la part de l'algorithme de détection. On a alors:

$$\begin{aligned} P_{fa} &= \sum_{y\in\Omega} \Pr(y) \\ P_{det} &= \sum_{y\in\Omega} \Pr(y|\,\mathcal{H}) \end{aligned}$$

(les sommes sont éventuellement à remplacer par des intégrales et les probabilités par des densités de probabilité dans le cas d'ensembles non discrets)

Le problème consiste à choisir la région Ω de manière à optimiser simultanément les deux probabilités. Or que l'on veuille maximiser P_{det} pour une valeur donnée de P_{fa} ou minimiser P_{fa} pour une valeur donnée de P_{det}, la meilleure stratégie consiste à intégrer dans Ω les éléments y de Y tels que le rapport $\frac{\Pr(y|\,\mathcal{H})}{\Pr(y)}$ soit supérieur à un certain seuil; la valeur de ce seuil étant choisie de manière à tomber sur celle des deux probabilités que l'on s'était donnée.

La fonction F idéale calcule donc ce rapport et le seuil S est celui dont on vient de parler.

Remarque 4 *Si F calcule le rapport $\frac{\Pr(y|\,\mathcal{H})}{\Pr(y)}$ et si le critère de détection est $F(y) \geq S$, on a alors $\frac{P_{det}}{P_{fa}} \geq S$, et en particulier $P_{fa} < 1/S$.*

Remarque 5 *L'ensemble Ω est décroissant (au sens de l'inclusion) par rapport au seuil S et tend vers l'ensemble vide. Si l'on exprime P_{det} en fonction de S, on obtient donc une fonction décroissante, tendant vers 0. Tenant compte de la remarque 4 et de ce qui est dit en fin de section précédente sur la discriminance, cette décroissance est d'autant plus lente que le critère est discriminant, et peut représenter une caractérisation de la discriminance.*

Les probabilité de détection et de fausse alarme dépendant du seuil S, on définira β et γ les fonctions de $\mathbf{R}^+$ dans $\mathbf{R}^+$ telles que $P_{det} = \beta(S)$ et $P_{fa} = \gamma(S)$.

Soit P_{fa}^{max} la probabilité de fausse alarme maximale que l'on s'autorise, et P_{det}^{min} un objectif que l'on aimerait atteindre pour la probabilité de détection. Si $\beta\left(\frac{1}{P_{fa}^{max}}\right) \geq P_{det}^{min}$, tout va bien, on pourra prendre $S = \frac{1}{P_{fa}^{max}}$ et l'objectif sera atteint. Si ça n'est pas le cas, on choisira le plus petit seuil S tel que $\gamma(S) \leq P_{fa}^{max}$.

Si l'on sait calculer $\beta(S)$ et non $\gamma(S)$ et si l'on a $\beta\left(\frac{1}{P_{fa}^{max}}\right) < P_{det}^{min}$, on considèrera que l'on a pour tout S, $\gamma(S) \leq \frac{\beta(S)}{S}$ et donc que le seuil S tel que $\frac{\beta(S)}{S} = P_{fa}^{max}$ convient. Il est inférieur à $\frac{1}{P_{fa}^{max}}$, on améliore donc la situation,

mais l'objectif pour la probabilité de détection ne sera cependant pas forcément atteint pour autant.

Si l'on ne sait calculer ni $\beta(S)$, ni $\gamma(S)$, ces considérations sont inutilisables. Toutefois, il est assez fréquent dans ce cas, que les deux quantités E et V que je définis plus bas, qui ne dépendent pas de S et qui permettent de faire des considérations analogues, soient caculables.

Supposons que le critère soit $F(y) \geq S$, que l'on ne sache pas calculer les probabilités de détection et de fausse alarme sous-jacentes, mais que l'on sache calculer:

$$E = \mathrm{Esp}(F(y)|\,\mathcal{H}) \qquad V = \mathrm{Var}(F(y)|\,\mathcal{H})$$

Si $E < \frac{1}{P_{fa}^{max}}$, alors le seuil $S = \frac{1}{P_{fa}^{max}}$ ne permettra probablement pas d'atteindre une probabilité de détection satisfaisante. On a en effet:

$$\beta\left(\frac{1}{P_{fa}^{max}}\right) = \Pr\left(F(y) \geq \frac{1}{P_{fa}^{max}}|\,\mathcal{H}\right) = \Pr\left(F(y) \Leftrightarrow E \geq \frac{1}{P_{fa}^{max}} \Leftrightarrow E|\,\mathcal{H}\right)$$

Et on pourra majorer la probabilité de détection en invoquant une inégalité de Chebyshev.

Bien sûr, tout couple (F', S') tel que:

$$\forall y \in Y \quad (F(y) \geq S) \Leftrightarrow (F'(y) \geq S')$$

peut remplacer (F, S) et est même préférable si F' est moins coûteuse à calculer.

Si l'on exige que ceci soit vérifié pour tout S, on a alors nécessairement une fonction $l : \mathbf{R}^+ \to \mathbf{R}^+$ strictement croissante telle que $F' = l \circ F$. Si cette fonction est continue, elle est alors bijective et l'on a $F = l^{-1} \circ F'$ et $F(y) \geq S \Leftrightarrow F'(y) \geq l(S)$.

Si l'on sait calculer $E' = \mathrm{Esp}(F'(y)|\,\mathcal{H})$ et $V' = \mathrm{Var}(F'(y)|\,\mathcal{H})$, et si $E' < l\left(\frac{1}{P_{fa}^{max}}\right)$, on aura de même que ci-dessus:

$$\beta\left(\frac{1}{P_{fa}^{max}}\right) = \Pr\left(F'(y) \geq l\left(\frac{1}{P_{fa}^{max}}\right)|\,\mathcal{H}\right) = \Pr\left(F'(y) \Leftrightarrow E' \geq l\left(\frac{1}{P_{fa}^{max}}\right) \Leftrightarrow E'\right)$$

Et l'on pourra de même invoquer une inégalité de Chebyshev pour majorer P_{det}. On trouvera les inégalités de Chebyshev usuelles, leur démonstration et leurs généralisations dans [Gal68], p. 126. Cela peut donner par exemple:

$$\forall S \qquad \beta(S) = \Pr(F'(y) \geq l(S)) \leq \frac{E'}{l(S)}$$

$$\forall S > l^{-1}(E') \qquad \beta(S) = \Pr(F'(y) \Leftrightarrow E' \geq l(S) \Leftrightarrow E') \leq \frac{V'}{(l(S) \Leftrightarrow E')^2}$$

Soit alors $\bar{\beta}(S)$ (< 1) une majoration calculable de $\beta(S)$, on a $P_{fa} \leq \frac{\bar{\beta}(S)}{S}$, et l'on pourra choisir le seuil S tel que $\frac{\bar{\beta}(S)}{S} = P_{fa}^{max}$, qui est calculable et inférieur à $\frac{1}{P_{fa}^{max}}$.

Nous verrons dans le prochain chapitre une application de la détection optimale. Elle permettra de détecter le type le plus simple de codage linéaire et de reconnaître la longueur et la synchronisation des mots de code.

2.1.4 Détection optimale sous contrainte de temps

Ains que nous le verrons dans le chapitre 3, nous pouvons appliquer un critère de détection optimal pour la détection d'un type de codage donné, si celui-ci est simple.

Pour la détection d'un code quelconque, l'hypothèse à vérifier: « un code a été utilisé » constitue une multitude (exponentielle en n^2) d'hypothèses simples du type « tel code a été utilisé ».

La situation peut se schématiser ainsi: on a cette fois:

$$A : \emptyset \rightarrow , , \qquad A' : , \ \rightarrow Y$$

Et l'on désire détecter, sur observation de $y \in Y$, le phénomène $A \circ A'$.

On dispose pour cela d'un modèle probabiliste décrivant de manière pertinente les phénomènes A, A' et l'évènement fortuit; c.a.d. que l'on est en mesure (encore que cela ne soit pas si sûr) de calculer pour tout $C \in$, et tout $y \in Y$ les probabilités:

$$\Pr(C|A), \qquad \Pr(y|C, A'), \qquad \Pr(y)$$

La probabilité $\Pr(y|A \circ A')$, quant à elle, vaut:

$$\Pr(y|A \circ A') = \sum_{C \in \Gamma} \Pr(C|A) \Pr(y|C, A')$$

Mais faire cette sommation est hors de portée, et l'on ne connaît pas d'autres manières d'évaluer cette probabilité. Le critère $\frac{\Pr(y|A \circ A')}{\Pr(y)}$ ne peut donc être appliqué.

Il paraît alors raisonnable de dire que sous la contrainte du temps (d'une puissance de calcul limité), on est forcé d'adopter une solution sous-optimale et que la fonction que l'on calculera ne sera pas égale au ratio $\frac{\Pr(y|A,A')}{\Pr(y)}$... Formellement, cela signifie, si l'on appelle $B : Y \rightarrow W$ notre fonction calculable avec la puissance de calcul impartie, que l'on rajoute un troisième niveau au phénomène à détecter: sur observation de $w \in W$, on veut distinguer le phénomène $A \circ A' \circ B$ du phénomène B (évènement fortuit suivi de B).

Compte tenu de ces éléments, le critère optimal consiste à comparer à un seuil la quantité:

$$\frac{\Pr(w|A \circ A' \circ B)}{\sum_{y \in Y} \Pr(y) \Pr(w|y, B)} = \frac{\Pr(w|A \circ A' \circ B)}{\Pr(w|B)}$$

Par exemple, pour le critère du rang, vu précédemment, on avait $B : Y \rightarrow \{\text{plein, non plein}\}$, $\Omega = \{\text{non plein}\}$, et:

$$\frac{\Pr(\text{non plein}|A \circ A' \circ B)}{\Pr(\text{non plein})} \approx 1 + 2^{-n} \sum_{i=1}^{n} C_i^{\perp} (1 + z^i)^N$$

(qui est égal à $\frac{P_{det}}{P_{fa}}$ dans ce cas car Ω est de cardinal 1)

Mais nous avons vu que la probabilité de détection allait en général être trop faible. Nous allons voir dans la suite que, plutôt que d'imposer $hX^t = 0$, il est plus intéressant de demander à hX^t d'être de poids faible. Et puisque nous nous plaçons dans le cas (fréquent) où le critère du rang a échoué, nous pourrons supposer que X est de rang plein. Mais n'anticipons pas...

On attend donc de la fonction B: qu'elle soit calculable bien-sûr, que le critère $\frac{\Pr(w|A\circ A'\circ B)}{\Pr(w|B)}$ le soit également, et surtout qu'il donne lieu à des probabilités de détection et de fausse alarme acceptables. Par ailleurs, nous aurons intérêt, nous allons voir pourquoi, à ce que cette fonction soit non déterministe.

Définition 12 *On appelle fonction non déterministe toute fonction dont le résultat ne dépend pas seulement de son argument, mais également du tirage aléataoire (dont elle se charge) d'un certain nombre de bits. Elle pourra donc donner des résultats différents lors d'appels successifs avec le même argument.*

Soit, pour tout $w \in W$, les probabilités $P_d(w) = \Pr(w|A\circ A'\circ B)$ et $P_f(w) = \Pr(w|B)$, et soit $\Omega(S)$ l'ensemble $\{w \in W / \frac{P_d(w)}{P_f(w)} > S\}$. Le critère de détection consiste à répondre positivement si $B(y) \in \Omega(S)$. La probabilité de détection vaut $\sum_{w\in\Omega(S)} P_d(w)$ et la probabilité de fausse alarme $\sum_{w\in\Omega(S)} P_f(w)$. Pour tout S, si ces probabilités ne sont pas nulles, leur ratio est supérieur à S.

Il arrive cependant que pour les valeurs acceptables de ce ratio, la probabilité de détection soit trop faible. Il est alors crucial que la fonction B soit non déterministe (contrairement au rang par exemple) et que l'on puisse l'appeler un grand nombre de fois, disons m, et pouvoir faire l'hypothèse que les résultats obtenus successivement soit distribués identiquement et indépendamment [1]. Le critère consiste alors à répondre positivement si pour au moins $x \geq 1$ des m appels, on a $B(y) \in \Omega(S)$.

Les paramètres à régler sont alors le nombre d'appels m et les seuils x et S. Nous allons voir que ceci peut nous offrir le grand avantage de choisir librement les probabilités de détection et de fausse alarme.

Appelons $P_{det}(S,m,x)$ et $P_{fa}(S,m,x)$ les probabilités de détection et de fausse alarme associées à un nombre m d'appels de la fonction B avec détection si et seulement si au moins x de ces m appels échouent dans l'ensemble $\Omega(S)$. On a en particulier $P_{det}(S,1,1) = \sum_{w\in\Omega(S)} P_d(w)$ et $P_{fa}(S,1,1) = \sum_{w\in\Omega(S)} P_f(w)$. Nous noterons $P_{det}(S,1,1) = \beta(S)$ et $P_{fa}(S,1,1) = \gamma(S)$.

Proposition 13 *Si les sorties successives de l'algorithme B sont distribuées identiquement et indépendamment, alors quels que soient $P_{det}^{min} \in]0,1[$, $P_{fa}^{max} \in]0,1[$ et $1 \leq x \leq m$, on a:*

$$\left(\sum_{i=0}^{x-1} \binom{m}{i} \beta(S)^i (1 \Leftrightarrow \beta(S))^{m-i} \leq 1 \Leftrightarrow P_{det}^{min}\right) \Rightarrow P_{det}(S,m,x) \geq P_{det}^{min}$$

1. « iidness »

$$\left(S \geq \frac{\binom{m}{x}^{1/x} \beta(S)}{(P_{fa}^{max})^{1/x}} \right) \Rightarrow P_{fa}(S, m, x) < P_{fa}^{max}$$

Démonstration: La probabilité $P_{det}(S, m, x)$ est la probabilité qu'au moins x des m appels à B donne $w \in \Omega(S)$ si $\mathcal{H}$ est vraie. $1 \Leftrightarrow P_{det}(S, m, x)$ est donc la probabilité, si $\mathcal{H}$ est vraie, qu'au plus $x \Leftrightarrow 1$ de ces appels donne $w \in \Omega(S)$. On a donc, du fait de l'indépendance des itérations:

$$1 \Leftrightarrow P_{det}(S, m, x) = \sum_{i=0}^{x-1} \binom{m}{i} \beta(S)^i (1 \Leftrightarrow \beta(S))^{m-i}$$

Si cette quantité est inférieure à $1 \Leftrightarrow P_{det}^{min}$ on aura donc $P_{det}(S, m, x) \geq P_{det}^{min}$.

Par ailleurs, par la borne de l'union, on a:

$$P_{fa}(S, m, x) < \binom{m}{x} \gamma(S)^x$$

Et puisque $\gamma(S) \leq \frac{\beta(S)}{S}$, on déduit la deuxième implication. □

On choisira donc S, m et x de manière à ce que ces inégalités soient vérifiées, et de manière aussi à minimiser m qui déterminera le coût de l'algorithme. Quelle est alors la plus petite valeur de m telle que ces inégalités soient vérifiées ? Que doivent valoir x et S ? Ces questions ne sont pas aussi simples qu'il n'y paraît. Les réponses dépendent de la fonction $\beta(S)$. On sera en général obligé de calculer numériquement le nombre m minimal pour chaque valeur de x, jusqu'à ce que ce nombre commence à augmenter avec x. Voici toujours, afin de mieux appréhender les dépendances antagonistes qui sont en jeu, un calcul formel de ce nombre pour $x = 1$ et 2:

Corollaire 2

$$\begin{cases} S = \frac{-\ln(1-P_{det}^{min})}{P_{fa}^{max}} \\ \beta(S) > 0 \\ m = \frac{-\ln(1-P_{det}^{min})}{\beta(S)} \end{cases} \Rightarrow \begin{cases} P_{det}(S, m, 1) > P_{det}^{min} \\ P_{fa}(S, m, 1) < P_{fa}^{max} \end{cases}$$

Démonstration: Par application de la proposition pour $x = 1$, on a:

$$(1 \Leftrightarrow \beta(S))^m \leq 1 \Leftrightarrow P_{det}^{min} \Rightarrow P_{det}(S, m, 1) \geq P_{det}^{min}$$

$$S \geq \frac{m\beta(S)}{P_{fa}^{max}} \Rightarrow P_{fa}(S, m, 1) < P_{fa}^{max}$$

Or, si $\beta(S) > 0$, $(1 \Leftrightarrow \beta(S))^m < e^{-m\beta(S)}$, donc

$$\begin{cases} \beta(S) > 0 \\ m = \frac{-\ln(1-P_{det}^{min})}{\beta(S)} \end{cases} \Rightarrow P_{det}(S,m,1) \geq P_{det}^{min}$$

Mais l'on a aussi dans ce cas $m\beta(S) = \Leftrightarrow \ln(1 \Leftrightarrow P_{det}^{min})$, il suffit donc de prendre $S = \frac{-\ln(1-P_{det}^{min})}{P_{fa}^{max}}$ pour avoir $P_{fa}(S,m,1) < P_{fa}^{max}$. □

Corollaire 3

$$\left(\exists a > 1 \quad \begin{cases} a \Leftrightarrow \ln a \geq \ln 2 \Leftrightarrow \ln(1 \Leftrightarrow P_{det}^{min}) \\ S = \frac{a+\frac{1}{2}}{\sqrt{2P_{fa}^{max}}} \\ \beta(S) > 0 \\ m = 1 + \frac{a}{\beta(S)} \end{cases} \right) \Rightarrow \begin{cases} P_{det}(S,m,2) > P_{det}^{min} \\ P_{fa}(S,m,2) < P_{fa}^{max} \end{cases}$$

Démonstration: Par application de la proposition pour $x = 2$, on a:

$$(1 \Leftrightarrow \beta(S))^m + m\beta(S)(1 \Leftrightarrow \beta((S))^{m-1} \leq 1 \Leftrightarrow P_{det}^{min} \Rightarrow P_{det}(S,m,2) \geq P_{det}^{min}$$

$$S \geq \frac{\sqrt{\frac{m(m-1)}{2}}\beta(S)}{\sqrt{P_{fa}^{max}}} \Rightarrow P_{fa}(S,m,2) < P_{fa}^{max}$$

Or, si l'on note $a = (m \Leftrightarrow 1)\beta(S)$ et si l'on suppose $a > 1$, on a:

$$(1 \Leftrightarrow \beta(S))^m + m\beta(S)(1 \Leftrightarrow \beta((S))^{m-1} = (1+a)(1 \Leftrightarrow \beta(S))^{m-1} < (1+a)e^{-a} < 2ae^{-a}$$

$$\text{Donc} \begin{cases} \beta(S) > 0, a > 1 \\ a \Leftrightarrow \ln a \geq \ln 2 \Leftrightarrow \ln(1 \Leftrightarrow P_{det}^{min}) \\ m = 1 + \frac{a}{\beta(S)} \end{cases} \Rightarrow P_{det}(S,m,2) \geq P_{det}^{min}.$$

Par ailleurs, puisque $m \Leftrightarrow \frac{1}{2} > \sqrt{m(m \Leftrightarrow 1)}$, on a $\frac{a+\frac{1}{2}}{\sqrt{2P_{fa}^{max}}} > \frac{\sqrt{\frac{m(m-1)}{2}}\beta(S)}{\sqrt{P_{fa}^{max}}}$.
il suffit donc de prendre $S = \frac{a+\frac{1}{2}}{\sqrt{2P_{fa}^{max}}}$ pour avoir $P_{fa}(S,m,2) < P_{fa}^{max}$. □

Je répète ici la remarque 5 de la section 2.1.3: La fonction $\beta(S)$ est décroissante, et tend vers 0. La décroissance est d'autant plus lente que le critère est discriminant.

La proposition 13 indique donc, mais cela se voit mieux sur ses corollaires, que m pourra être d'autant plus petit que le critère est discriminant.

Nous appliquerons ces résultats à la détection de code au chapitre 4, la fonction B sera liée à la génération de mots h tels hX^t soit de poids faible. Les principes de la détection optimale sous contrainte de temps permettront de dimensionner correctement les paramètres des algorithmes de détection.

2.2 Problèmes de reconnaissance

Les problèmes de reconnaissance ressemblent un peu aux problèmes de détection mais ne peuvent toutefois se formaliser ni se traiter de la même manière. Il faut souvent les aborder sous l'angle d'une «reconstruction», où un candidat se rapproche pas à pas de l'objet à reconnaître.

Le phénomène considéré n'est plus $A : \emptyset \to Y$ mais $A : S \to Y$ où S est l'ensemble des « sources » possibles du phénomène. Il ne s'agit plus de distinguer A de l'évènement fortuit mais de reconnaître, sur observation de $y \in Y$, la source $s \in S$ la plus vraisemblable, c.a.d. celle qui maximise $\Pr(s)\Pr(y|s)$.

Par hypothèse, on est assuré que A a eu lieu et le concept de fausse alarme n'a pas lieu d'être. Le danger à éviter est cette fois la mauvaise reconnaissance.

Contrairement aux problèmes de détection qui ont beaucoup de points communs entre eux, les différents problèmes de reconnaissance se traitent spécifiquement et l'on n'a pas beaucoup de généralités à dire sur ce sujet. Nous parlons ci-dessous des problèmes de reconnaissance spécifiques qui nous intéressent.

2.2.1 Reconnaissance optimale

Pour une sortie $y \in Y$ donnée, la source $s \in S$ la plus vraisemblable est celle qui maximise la quantité $\Pr(s)\Pr(y|s)$. Nous allons au prochain chapitre appliquer la reconnaissance optimale à la reconnaissance de longueur et de synchronisation dans le cas d'un type de codage fort simple.

Il s'agira de trouver la longueur n et la synchronisation s qui maximise la quantité $\Pr(n,s)\Pr(y|n,s)$ que nous saurons calculer et il sera possible d'essayer toutes les hypothèses de longueur et de synchronisation.

2.2.2 Reconnaissance sous contrainte de temps

Si le type de codage est inconnu et s'il s'agit de trouver la longueur n et la synchronisation s qui maximise la quantité $\Pr(n,s)\Pr(y|n,s)$, il sera possible d'essayer toutes les hypothèses de longueur et de synchronisation, mais nous verrons qu'il n'est en général pas envisageable de calculer, pour n et s donnés, la valeur exacte de $\Pr(y|n,s)$. Il s'agira donc de construire une fonction f (pas nécessairement déterministe), que nous appellerons «estimation de vraisemblance», qui doit être « aussi monotone que possible » en $\Pr(n,s)\Pr(y|n,s)$ afin de sélectionner l'hypothèse la plus vraisemblable.

Pour la reconnaissance de code, on considèrera que la longueur et la synchronisation sont connues et il s'agit de trouver le code C qui maximise $\Pr(C)\Pr(y|C)$. La difficulté cette fois est double: d'une part S, l'ensemble de tous les codes envisageables est en général trop grand (exponentiel en n^2) pour être parcouru exhaustivement, et d'autre part, pour un code C donné, la probabilité $\Pr(y|C)$ est trop complexe à calculer exactement.

Il faudra donc cette fois construire un générateur d'hypothèses (du type « tel code a été utilisé ») ayant de fortes chances de générer la bonne hypothèse, en plus d'une fonction f de type «estimation de vraisemblance».

Conclusion

Nous savons maintenant ce que les algorithmes de détection doivent calculer. Nous allons en mesurer l'utilité dans les deux prochains chapitres.

Nous verrons dans le prochain chapitre que la détection et la reconnaissance optimale vont nous permettre de résoudre le problème de manière optimale si on en simplifie l'énoncé.

Dans le suivant, l'énoncé général et la contrainte de temps rendront impossible cette optimalité. Nous verrons alors qu'il faudra se fier à la sortie de sous-routines pertinemment choisies. La notion de discriminance sera alors fondamentale et nous aidera à faire ce choix. Et les principes de la détection optimale sous contrainte de temps nous permettront d'exploiter au mieux les réponses de ces sous-routines.

Les diverses considérations que nous avons faites seront indispensables pour dimensionner correctement les paramètres des différents algorithmes que nous allons définir.

Chapitre 3

Exemple de détection optimale: Code de parité [n,n-1]

La raison nous trompe plus souvent que la nature.
Marquis de Vauvenargues (Réflexions et maximes)

Il est curieux de constater - et la relativité n'est pas la seule à nous le montrer - qu'à mesure que le raisonnement progresse, il prétend de moins en moins être à même de prouver.
Bertrand Russel (ABC de la relativité)

La foi du savant ne ressemble pas à celle que les orthodoxes puisent dans le besoin de certitude. Il ne faut pas croire que l'amour de la vérité se confonde avec celui de la certitude... *Henri Poincaré*

Introduction

Dans ce chapitre, nous étudions un algorithme de détection lié à l'hypothèse que la transmission a été supportée par un code de parité $[n, n \Leftrightarrow 1]$ (on rajoute un bit de parité à chaque $(n \Leftrightarrow 1)\Leftrightarrow$uplets de bits d'information).

Nous supposerons tout d'abord que la longueur n de l'hypothétique codage est connue, ainsi que la synchronisation, c.a.d. que l'on sait découper le train binaire en mots qui, si l'hypothèse est juste, sont des mots de code

Dans la section 3.3, p. 128, « Reconaissance de longueur et de synchronisation », nous abandonnons cette hypothèse et étudions le nouveau problème de détection qui se pose.

Les solutions qui sont proposées sont optimales dans les deux cas, si tant est que l'on puisse parler d'optimalité lorsqu'est requise une connaissance parfaite des diverses probabilités régissant les sources possibles du train binaire et que cette connaissance ne peut qu'être approximative (sans quoi d'ailleurs le problème de détection n'aurait pas lieu d'être). Par ailleurs, les hypothèses de travail que nous choisirons seront aussi générales que possible et des solutions meilleures pourraient se présenter s'il s'agissait de traiter une plus grande spécificité.

3.1 Cas dur

3.1.1 Principe de l'algorithme de détection

Les mots de code émis sont caractérisés par le fait que leur poids de Hamming est pair. Il est vrai que cette parité peut être modifiée par les erreurs de transmission, mais il n'y a pas d'autre méthode de détection que de mesurer la proportion de mots de poids pair ou impair parmi les mots de l'espace ambiant que constituent les tronçons de longueur n du train binaire intercepté.

L'algorithme décidera que l'hypothèse est juste si la proportion de poids impair est inférieure à un seuil donné, fonction des probabilités de détection et de fausse alarme requises.

3.1.2 Variables aléatoires

L'hypothèse à vérifier, que nous noterons $\mathcal{H}$, est que les mots transmis sont tous de poids pair. Nous supposerons que si l'hypothèse est fausse, les mots transmis sont absolument quelconques.

Nous considérons les trois variables aléatoires suivantes:

- Le mot de code transmis, élément (si $\mathcal{H}$ est vraie) de l'ensemble C des mots binaires de longueur n et de poids pair, est une variable aléatoire γ. Par hypothèse, la distribution est homogène, c.a.d. que tous les élément de C ont la même probabilité d'être transmis:

$$\forall c \in C \qquad \Pr(\gamma = c | \mathcal{H}) = 2^{1-n}$$

- L'erreur de transmission, mot binaire de longueur n, est une variable aléatoire δ, dont la distribution de probabilité dépend du canal. Nous considérons un canal binaire symétrique, sans mémoire, et de probabilité de transition τ, nous avons donc:

$$\forall e \in \mathbf{F}_2^n \qquad \Pr(\delta = e) = \tau^{\mathrm{wt}(e)}(1 \Leftrightarrow \tau)^{n-\mathrm{wt}(e)}$$

- Le mot « reçu », somme du mot de code transmis et de l'erreur de transmission, est une variable aléatoire λ. Nous noterons Λ_N le produit combinatoire de N variables aléatoires λ, représenté par une $N \times n$⇔matrice binaire. Sa distribution de probabilité est définie par celle de γ et δ.

L'observation dont disposera l'algorithme de détection pour prendre sa décision sera une réalisation X de la variable aléatoire Λ_N. On a vu dans le chapitre précédent traitant des généralités sur les algorithmes de détection que le critère de détection doit consister à comparer $\frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)}$ à un seuil S.

Nous allons voir dans la prochaine section que ceci revient précisément à comparer la proportion de mots de poids impair parmi les mots reçus à une proportion seuil.

3.1.3 Calcul des probabilités

Nous noterons $\bar{1}$ le mot de l'espace ambiant composé uniquement de 1 (c.a.d. le mot non nul du dual du code de parité), X^t la transposée d'une matrice X et $\langle x, y\rangle (= xy^t)$ le produit scalaire de deux vecteurs x et y.

Nous faisons référence ci-dessous au lemme 3, p. 99, et nous reprenons la notation $z = 1 \Leftrightarrow 2\tau$.

Proposition 14 *Si $\mathcal{H}$ est vraie, la probabilité pour que le poids de Hamming d'un mot reçu soit pair vaut:*

$$\Pr(\langle \lambda, \bar{1}\rangle = 0|\,\mathcal{H}) = \frac{1 + z^n}{2}$$

En l'absence d'hypothèse en revanche, on a:

$$\Pr(\langle \lambda, \bar{1}\rangle = 0) = \frac{1}{2}$$

Démonstration: La deuxième partie de la proposition est triviale. La première partie se démontre de la manière suivante:

Si $\mathcal{H}$ est vrai, le poids du mot transmis est pair. Le poids du mot reçu sera donc pair si et seulement si l'erreur de transmission est elle-même de poids pair, ce qui arrive avec une probabilité:

$$\sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{2i} \tau^{2i} (1 \Leftrightarrow \tau)^{n-2i}$$

La proposition se déduit donc du lemme 3, p. 99. □

Proposition 15

$$\forall x \in \mathbf{F}_2^n \qquad \frac{\Pr(\lambda = x|\,\mathcal{H})}{\Pr(\lambda = x)} = 1 + (\Leftrightarrow 1)^{\langle x, \bar{1}\rangle} z^n$$

Démonstration: En l'absence d'hypothèse, les 2^n mots de l'espace ambiant sont equiprobables. On a donc $\Pr(\lambda = x) = 2^{-n}$. Sous l'hypothèse $\mathcal{H}$ en

revanche, on a:

$$\begin{aligned}\Pr(\lambda = x|\,\mathcal{H}) &= \sum_{c\in C}\Pr(\gamma = c)\Pr(\delta = x+c)\\ &= 2^{1-n}\sum_{c\in C}\Pr(\delta = x+c)\\ &= \begin{cases}2^{1-n}\frac{1+z^n}{2} & \text{si } \langle x,\bar{1}\rangle = 0,\\ 2^{1-n}\frac{1-z^n}{2} & \text{si } \langle x,\bar{1}\rangle = 1.\end{cases}\end{aligned} \tag{3.1}$$

□

Corollaire 4

$$\forall X \in \mathbf{F}_2^{N\times n} \qquad \frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)} = (1+z^n)^N\left(\frac{1\Leftrightarrow z^n}{1+z^n}\right)^{wt(\bar{1}.X^t)}$$

Démonstration: On suppose bien-sûr que les mots successivement transmis sont statistiquement indépendants et l'on a alors en appelant $X_1 \dots X_N$ les lignes de la matrice X:

$$\frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)} = \prod_{i=1}^{N}\frac{\Pr(\lambda = X_i|\,\mathcal{H})}{\Pr(\lambda = X_i)}$$

On déduit le corollaire du fait que $\mathrm{wt}(\bar{1}.X^t)$ représente le nombre de mots de poids impair parmi les lignes de la matrice X. □

On a vu dans le chapitre traitant des généralités sur les algorithmes de détection que le critère de détection devait consister à comparer $\frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)}$ à un seuil S. Or on a bien-sûr, compte tenu de la dernière proposition, quels que soient S et X:

$$\frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)} \geq S \Leftrightarrow wt(\bar{1}.X^t) \leq \frac{N\log(1+z^n)\Leftrightarrow\log S}{\log(1+z^n)\Leftrightarrow\log(1\Leftrightarrow z^n)}$$

L'algorithme comparera donc, ainsi que l'on s'en doutait, la proportion de mots de poids impair parmi les mots reçus à une proportion seuil α préalablement calculé et décidera que $\mathcal{H}$ est vraie en cas d'infériorité.

Remarquons que ce seuil vaut $\alpha = \frac{1}{1-\frac{\log(1-z^n)}{\log(1+z^n)}} \Leftrightarrow \frac{\frac{S}{N}}{\log(1+z^n)-\log(1-z^n)}$. Or $\frac{-\log(1-z^n)}{\log(1+z^n)} > 1$, et donc $\alpha < \frac{1}{2}$. Par ailleurs, quel que soit le seuil S choisi, α tend vers $\frac{1}{1-\frac{\log(1-z^n)}{\log(1+z^n)}}$ pour N tendant vers l'infini.

Pour minorer cette limite, il nous faut minorer $\log(1\Leftrightarrow x)$. Enonceons donc ce petit théorème. La base du logarithme est la base naturelle.

Théorème 11

$$\forall x \in]0,1[\qquad \forall i \geq 2 \qquad \log(1\Leftrightarrow x) > \Leftrightarrow\sum_{j=1}^{i-1}\frac{x^j}{j} \Leftrightarrow \frac{x^i}{i(1\Leftrightarrow x)}$$

Démonstration: Il est equivalent de démontrer que $(1 \Leftrightarrow x)\log(1 \Leftrightarrow x) > \Leftrightarrow(1 \Leftrightarrow x)\sum_{j=1}^{i-1} \frac{x^j}{j} \Leftrightarrow \frac{x^i}{i}$. Soit $f(x)$ et $g(x)$ les membres de gauche et de droite respectivement de cette inéquation. On a $f(0) = g(0) = 0$. Dérivons f et g:

$$f'(x) = \Leftrightarrow\log(1 \Leftrightarrow x) \Leftrightarrow 1$$

$$g'(x) = \sum_{j=1}^{i-1} \frac{x^j}{j} \Leftrightarrow (1 \Leftrightarrow x) \sum_{j=1}^{i-1} x^{j-1} \Leftrightarrow x^{i-1} = \sum_{j=1}^{i-1} \frac{x^j}{j} \Leftrightarrow 1$$

Or, pour $x \in]0,1[$, il est bien connu que $\log(1 \Leftrightarrow x) < \Leftrightarrow\sum_{j=1}^{i-1} \frac{x^j}{j}$, on a donc $f'(x) > g'(x)$, donc $f(x) > g(x)$. □

Corollaire 5 *En majorant* $\log(1 + z^n)$ *par* z^n*, et en utilisant le théorème pour* $i = 2$*, on trouve que:*

$$\frac{1}{1 \Leftrightarrow \frac{\log(1-z^n)}{\log(1+z^n)}} > \frac{1 \Leftrightarrow z^n}{2 \Leftrightarrow \frac{3}{2}z^n} > \frac{1 \Leftrightarrow z^n}{2}$$

Le critère de détection étant défini, on en déduit les probabilités de détection et de fausse alarme:

$$\begin{aligned} P_{det}(\alpha, N) &= \Pr(wt(\bar{1}.\Lambda_N^t) \leq \alpha N | \mathcal{H}) \\ P_{fa}(\alpha, N) &= \Pr(wt(\bar{1}.\Lambda_N^t) \leq \alpha N) \end{aligned}$$

L'espérance de $\langle \bar{1}, \lambda \rangle$ est $\frac{1}{2}$ en général, et $\frac{1-z^n}{2}$ si $\mathcal{H}$ est vraie. Les lignes λ de Λ_N étant identiques et indépendantes, la loi faible des grands nombres nous indiquent que si $\frac{1-z^n}{2} < \alpha < \frac{1}{2}$ alors:

$$\lim_{N \to \infty} P_{fa}(\alpha, N) = 0 \qquad \lim_{N \to \infty} P_{det}(\alpha, N) = 1$$

Ceci implique que, si l'on suppose que l'on n'est pas limité en temps de calcul ni en nombre de mots reçus (ce qui n'est bien sûr pas le cas mais la remarque est intéressante), on pourra faire tendre simultanément la probablité de détection vers 1 et la probabilité de fausse alarme vers 0.

La proposition suivante et son corollaire offrent davantage de précision:

Proposition 16

$$\begin{aligned} P_{det}(\alpha, N) &= 2^{-N} \sum_{i=0}^{\alpha N} \binom{N}{i} (1 \Leftrightarrow z^n)^i (1 + z^n)^{N-i} \\ P_{fa}(\alpha, N) &= 2^{-N} \sum_{i=0}^{\alpha N} \binom{N}{i} \end{aligned}$$

Corollaire 6 *Si* $\beta = \frac{1-z^n}{2}$ *et si* $\beta < \alpha < \frac{1}{2}$, *alors il existe* $K_1 < 1$ *et* $K_2 < 1$ *tels que, pour tout* N*:*

$$\begin{aligned} P_{fa}(\alpha, N) &\leq K_1^N \\ P_{det}(\alpha, N) &\geq 1 \Leftrightarrow \frac{\alpha(1 \Leftrightarrow \beta)}{\alpha \Leftrightarrow \beta} K_2^N \end{aligned}$$

Démonstration: On a:

$$P_{fa}(\alpha, N) = 2^{-N} \sum_{i=0}^{\alpha N} \binom{N}{i}$$

On en déduit ([MS77] p.310):

$$P_{fa}(\alpha, N) \leq \left(2^{H_2(\alpha)-1}\right)^N$$

où H_2 est la fonction entropie:

$$H_2(x) = \Leftrightarrow x \log_2(x) \Leftrightarrow (1 \Leftrightarrow x) \log_2(1 \Leftrightarrow x)$$

$\alpha < 1/2$ donc $H_2(\alpha) < 1$ et $2^{H_2(\alpha)-1} < 1$. On peut donc prendre $K_1 = 2^{H_2(\alpha)-1}$ pour obtenir la première partie du corollaire.

Par ailleurs:

$$P_{det}(\alpha, N) = \sum_{i=0}^{\alpha N} \binom{N}{i} \beta^i (1 \Leftrightarrow \beta)^{N-i}$$

et ainsi que l'on peut le vérifier dans [Gal68] (p. 530)

$$\sum_{i=0}^{\alpha N} \binom{N}{i} \beta^i (1 \Leftrightarrow \beta)^{N-i} \geq 1 \Leftrightarrow \frac{\alpha(1 \Leftrightarrow \beta)}{\alpha \Leftrightarrow \beta} \binom{N}{\alpha N} \beta^{\alpha N} (1 \Leftrightarrow \beta)^{(1-\alpha)N}$$

Mais $\binom{N}{\alpha N} \leq 2^{N H_2(\alpha)} = (\alpha^\alpha (1 \Leftrightarrow \alpha)^{1-\alpha})^{-N}$, donc

$$\binom{N}{\alpha N} \beta^{\alpha N} (1 \Leftrightarrow \beta)^{(1-\alpha)N} \leq \left(\left(\frac{\beta}{\alpha}\right)^\alpha \left(\frac{1 \Leftrightarrow \beta}{1 \Leftrightarrow \alpha}\right)^{1-\alpha} \right)^N$$

Soit $K_2 = (\frac{\beta}{\alpha})^\alpha (\frac{1-\beta}{1-\alpha})^{1-\alpha}$, on a $\ln(K_2) = \alpha \ln(\frac{\beta}{\alpha}) + (1 \Leftrightarrow \alpha) \ln(\frac{1-\beta}{1-\alpha})$.
Mais ($x \neq 1 \Rightarrow \ln(x) < x \Leftrightarrow 1$) et $\alpha \neq \beta$, on a donc:

$$\ln(K_2) < \alpha \left(\frac{\beta}{\alpha} \Leftrightarrow 1\right) + (1 \Leftrightarrow \alpha) \left(\frac{1 \Leftrightarrow \beta}{1 \Leftrightarrow \alpha} \Leftrightarrow 1\right) = 0$$

Donc $K_2 < 1$, et l'on obtient ainsi la deuxième partie du corollaire. □

Quoi qu'il en soit, si P_{fa}^{max} est la probabilité de fausse alarme maximale que l'on s'autorise, et étant donné N, on choisira le plus grand α tel que $P_{fa}(\alpha, N) \leq P_{fa}^{max}$. On pourra alors juger de la grandeur de $P_{det}(\alpha, N)$ mais on sera impuissant à l'augmenter.

3.2 Cas souple

On pourra relire la section 3.1 pour se remémorer les notations et leur signification.

3.2.1 Variables aléatoires

Nous considérerons les variables aléatoires suivantes:

- le bit transmis, élément de $\mathbf{F}_2$, sera représenté par une variable aléatoire β. Il vaut 0 ou 1 avec une probabilité 1/2.
- Le mot transmis, élément (si $\mathcal{H}$ est vraie) de l'ensemble C des mots binaires de longueur n et de poids pair, est une variable aléatoire γ. Par hypothèse, la distribution est homogène, c.a.d. que tous les élément de C ont la même probabilité d'être transmis:

$$\forall c \in C \qquad \Pr(\gamma = c | \mathcal{H}) = 2^{1-n}$$

- Le bruit, nombre réel, que l'on a déjà mentionné, est une variable aléatoire continue ϵ dont la densité de probabilité dépend du canal. Pour le canal sans mémoire, à bruit blanc, gaussien et additif de densité monolatérale $N0/2$, nous avons:

$$\forall e \in \mathbf{R} \qquad \rho(\epsilon = e) = \frac{1}{\sqrt{\pi N0}} \exp\left(\Leftrightarrow \frac{\mathcal{E}^2}{N0}\right)$$

- L'erreur de transmission, n⇔uplet de nombres réels, est le produit combinatoire de n variables aléatoires ϵ représenté par la variable aléatoire δ. Sa densité de probabilité est:

$$\forall d \in \mathbf{R}^n \qquad \rho(\delta = d) = (\pi N0)^{-n/2} e^{-\frac{\sum_{i=1}^{n} d_i^2}{N0}}$$

- Le signal « reçu », nombre réel, somme du signal transmis ($\pm\,\mathcal{E}$) et du bruit, est une variable aléatoire α. Sa densité de probabilité est définie par celles de β et ϵ. C'est le réel qui est mesuré par le démodulateur.
- Le mot « reçu », n⇔uplet de nombres réels, somme du mot transmis modulé et de l'erreur de transmission, est une variable aléatoire λ. Nous noterons Λ_N le produit combinatoire de N variables aléatoires λ, représenté par une $N \times n$⇔matrice réelle. Sa densité de probabilité est définie par celle de γ et δ.

L'observation dont disposera l'algorithme de détection pour prendre sa décision sera une réalisation L de la variable aléatoire Λ_N. On a vu dans le chapitre précédent traitant des généralités sur les algorithmes de détection que le critère de détection doit consister à comparer $\frac{\Pr(\Lambda_N = L | \mathcal{H})}{\Pr(\Lambda_N = L)}$ à un seuil S.

3.2.2 Calcul des probabilités

Pour $b \in \mathbf{F}_2$ et $a \in R$, nous noterons $\Pr(\beta = b|\alpha = a)$ la probabilité pour que le bit envoyé vaille b si le réel a est mesuré. Ainsi nous avons (les bits 0 ou 1 étant a priori équiprobable):

$$\Pr(\beta = 0|\alpha = a) = \frac{e^{\frac{-(a-\mathcal{E})^2}{N0}}}{e^{-\frac{(a-\mathcal{E})^2}{N0}} + e^{-\frac{(a+\mathcal{E})^2}{N0}}} = \frac{1}{1 + e^{\frac{-4a\,\mathcal{E}}{N0}}} = \frac{e^{\frac{2a\,\mathcal{E}}{N0}}}{2\cosh(\frac{2a\,\mathcal{E}}{N0})}$$

$$\Pr(\beta = 1|\alpha = a) = \frac{e^{\frac{-(a+\mathcal{E})^2}{N0}}}{e^{-\frac{(a-\mathcal{E})^2}{N0}} + e^{-\frac{(a+\mathcal{E})^2}{N0}}} = \frac{1}{1 + e^{\frac{4a\,\mathcal{E}}{N0}}} = \frac{e^{\frac{-2a\,\mathcal{E}}{N0}}}{2\cosh(\frac{2a\,\mathcal{E}}{N0})}$$

Pour alléger les notations, nous définissons la quantité $A = \frac{N0}{2\,\mathcal{E}}$. Ainsi, nous avons:

$$\Pr(\beta = 0|\alpha = a) = \frac{e^{a/A}}{2\cosh(a/A)} \qquad \Pr(\beta = 1|\alpha = a) = \frac{e^{-a/A}}{2\cosh(a/A)}$$

Proposition 17 *Pour tout n⇔uplets $(a_1 \dots a_n)$ de nombres réels, on a:*

$$\rho(\lambda = a_1 \dots a_n) = \frac{\exp\left(\Leftrightarrow\frac{\sum_{i=1}^n a_i^2 + n\,\mathcal{E}^2}{N0}\right)}{(\pi N0)^{n/2}} \prod_{i=1}^n \cosh\left(\frac{a_i}{A}\right)$$

$$\rho(\lambda = a_1 \dots a_n|\,\mathcal{H}) = \frac{\exp\left(\Leftrightarrow\frac{\sum_{i=1}^n a_i^2 + n\,\mathcal{E}^2}{N0}\right)}{(\pi N0)^{n/2}} \left(\prod_{i=1}^n \cosh\left(\frac{a_i}{A}\right) + \prod_{i=1}^n \sinh\left(\frac{a_i}{A}\right)\right)$$

Démonstration: Pour $b \in \mathbf{F}_2$ et $a \in \mathbf{R}$, la densité de probabilité de mesurer a si b est émis vaut $\rho(\alpha = a|\beta = b) = \frac{1}{\sqrt{\pi N0}} e^{-\frac{(a-(-1)^b\,\mathcal{E})^2}{N0}} = \frac{\exp\left(-\frac{a^2+\mathcal{E}^2}{N0}\right)}{\sqrt{\pi N0}} e^{(-1)^b a/A}$.

Dans le premier cas, les a_i sont indépendants, donc $\rho(\lambda = a_1 \dots a_n) = \prod_{i=1}^n \rho(\alpha = a_i)$.

Mais $\rho(\alpha = a) = \frac{1}{2}(\rho(a|0) + \rho(a|1)) = \frac{\exp\left(-\frac{a^2+\mathcal{E}^2}{N0}\right)}{\sqrt{\pi N0}} \cosh\left(\frac{a}{A}\right)$, d'où la première formule.

Dans le second cas, en revanche, on a:

$$\begin{aligned}
\rho(\lambda = a_1 \dots a_n|\,\mathcal{H}) &= \sum_{c \in C} \Pr(\gamma = c|\,\mathcal{H}) \rho(\lambda = a_1 \dots a_n|\gamma = c) \\
&= 2^{1-n} \sum_{c \in C} \prod_{i=1}^n \rho(\alpha = a_i|\beta = c_i) \\
&= \frac{\exp\left(\Leftrightarrow\frac{\sum_{i=1}^n a_i^2 + n\,\mathcal{E}^2}{N0}\right)}{2^{n-1}(\pi N0)^{n/2}} \sum_{c \in C} \exp\left(\sum_{i=1}^n (\Leftrightarrow 1)^{c_i} a_i/A\right)
\end{aligned}$$

Or $\prod_{i=1}^{n}\left(e^{\frac{a_i}{A}}+e^{-\frac{a_i}{A}}\right)=\sum_{b\in\mathbf{F}_2^n}\exp\left(\sum_{i=1}^{n}(\Leftrightarrow 1)^{b_i}a_i/A\right)$,

et $\prod_{i=1}^{n}\left(e^{\frac{a_i}{A}}\Leftrightarrow e^{-\frac{a_i}{A}}\right)=\sum_{b\in\mathbf{F}_2^n}(\Leftrightarrow 1)^{\mathrm{wt}(b)}\exp\left(\sum_{i=1}^{n}(\Leftrightarrow 1)^{b_i}a_i/A\right)$.

Donc, C étant composé des mots de poids pair, on a:

$$\sum_{c\in C}\exp\left(\sum_{i=1}^{n}(\Leftrightarrow 1)^{c_i}a_i/A\right)=\frac{\prod_{i=1}^{n}\left(e^{\frac{a_i}{A}}+e^{-\frac{a_i}{A}}\right)+\prod_{i=1}^{n}\left(e^{\frac{a_i}{A}}\Leftrightarrow e^{-\frac{a_i}{A}}\right)}{2}$$

$$\text{et}\quad\frac{\sum_{c\in C}\exp\left(\sum_{i=1}^{n}(\Leftrightarrow 1)^{c_i}a_i/A\right)}{2^{n-1}}=\prod_{i=1}^{n}\cosh\left(\frac{a_i}{A}\right)+\prod_{i=1}^{n}\sinh\left(\frac{a_i}{A}\right)$$

On en déduit la seconde formule. □

Corollaire 7

$$\forall(a_1\ldots a_n)\in\mathbf{R}^n\qquad\frac{\rho(\lambda=a_1\ldots a_n|\,\mathcal{H})}{\rho(\lambda=a_1\ldots a_n)}=1+\prod_{i=1}^{n}\tanh\left(\frac{a_i}{A}\right)$$

Corollaire 8 *Pour toute $N\times n$-matrice réelle L dont les coefficients sont notés $a_{i,j}$ $(1\leq i\leq N, 1\leq j\leq n)$ on a:*

$$\frac{\rho(\Lambda_N=L|\,\mathcal{H})}{\rho(\Lambda_N=L)}=\prod_{i=1}^{N}\left(1+\prod_{j=1}^{n}\tanh\left(\frac{a_{i,j}}{A}\right)\right)$$

L'algorithme de détection reçoit en entrée une matrice $L=(a_{i,j})$ comme définie ci-dessus. Notre critère de détection, paramétrable par un réel S, est donc défini par: Réponse affirmative si et seulement si $\prod_{i=1}^{N}\left(1+\prod_{j=1}^{n}\tanh\left(\frac{a_{i,j}}{A}\right)\right)>S$.

Les probabilités de fausse alarme et de détection valent alors:

$$P_{fa}(S)=\Pr\left(\prod_{i=1}^{N}\left(1+\prod_{j=1}^{n}\tanh\left(\frac{a_{i,j}}{A}\right)\right)>S\right)$$

$$P_{det}(S)=\Pr\left(\prod_{i=1}^{N}\left(1+\prod_{j=1}^{n}\tanh\left(\frac{a_{i,j}}{A}\right)\right)>S|\,\mathcal{H}\right)$$

Le probléme est que ces probabilités sont difficiles à calculer ou à approximer dès que N dépasse quelques unités (il s'agit d'intégrales non simplfiables sur une région de $\mathbf{R}^N$), et il faudra généralement choisir S de manière semi-empirique.

Considérons cependant la fonction $F'(L)=\sum_{i=1}^{N}\log\left(1+\prod_{j=1}^{n}\tanh\left(\frac{a_{i,j}}{A}\right)\right)$, c.a.d. le logarithme du critère. Et soient E' l'espérance et V' la variance de F' si $\mathcal{H}$ est vraie. E'/N et V'/N sont clairement indépendants de N et ne dépendent

que de la longueur n et du rapport signal à bruit $\mathcal{E}^2/N0$. On pourra donc calculer des abaques pour E'/N et V'/N, de manière à avoir directement accès (par interpolation et multiplication par N) à E' et V'. Pour tout seuil $S > e^{E'}$, on pourra alors faire les majorations suivantes, issues d'inégalités de Chebyshev (cf. par exemple [Gal68], p 126):

$$P_{det}(S) \leq \min\left(\frac{E'}{\log(S)}, \frac{V'}{(\log(S) \Leftrightarrow E')^2}\right)$$

$$P_{fa}(S) \leq \frac{P_{det}(S)}{S}$$

Si P_{fa}^{max} est la probabilité de fausse alarme maximale que l'on s'autorise, on pourra alors choisir le seuil S tel que $\frac{\min\left(\frac{E'}{\log(S)}, \frac{V'}{(\log(S)-E')^2}\right)}{S} = P_{fa}^{max}$.

3.3 Reconaissance de longueur et de synchronisation

Les deux sections précédentes supposent que l'on connaisse la longueur de l'hypothétique code de parité ainsi que la synchronisation (le début et la fin de chaque hypothétique mot de code).

Tel peut être le cas si par exemple une mise en trame est détectée auparavant sur le train binaire. La détection de mise en trame n'est pas abordée dans cette thèse mais il faut savoir que la plupart des protocoles de transmission comprennent de nombreuses sophistications quant au format des trains binaires devant transiter dans le canal, dont une utilité (parmi d'autres) est de permettre de récupérer la synchronisation (le train binaire contient même assez souvent en préfixe des informations sur sa nature et sur son encodage).

Nous supposerons maintenant qu'il n'en est plus ainsi mais que plusieurs hypothèses différentes sont émises. Si l'on est sûr d'avoir le train binaire en entier, les longueurs de code possibles sont les diviseurs de la longueur du train binaire; et pour chaque longueur, on aura une unique hypothèse de synchronisation.

Si en revanche on ne dispose que d'une partie (connexe) du train, toutes les longueurs sont théoriquement possibles. Si l'on a une extrémité de ce train (qu'il s'agisse du premier ou du dernier bit), une seule hypothèse de synchronisation par longueur suffit. Dans le cas contraire, pour chaque longueur n, n hypothèses de synchronisation sont nécessaires.

Selon les cas, on aura donc un certain nombre l d'hypothèses $\mathcal{H}_1 \ldots \mathcal{H}_l$ de longueur et de synchronisation, chacune pouvant être pondérée par une probabilité a priori que nous noterons $\Pr(\mathcal{H}_i \,|\, \mathcal{H})$ et que nous prendrons homogène (égale à $1/l$) par défaut. L'hypothèse $\mathcal{H}$ consiste à dire que l'une de ces hypothèses est vraie. Si l'algorithme décide que $\mathcal{H}$ est vraie, il lui faudra ensuite chercher laquelle de ces hypothèses est la plus probable. Il aura alors effectué ce que l'on a appelé la reconnaissance de longueur et de synchronisation.

3.3.1 Cas dur

Appelons X le train binaire effectivement intercepté (il ne s'agit plus d'une matrice mais d'un vecteur) et Λ la variable aléatoire correspondante. Et pour chaque hypothèse $\mathcal{H}_i$, appelons n_i la longueur de code correspondante, N_i le nombre de mots de code correspondant, X_i la $n_i \times N_i$-matrice extraite de X comme préconisé par l'hypothèse $\mathcal{H}_i$ et Λ_i la variable aléatoire correspondante. On a alors, pour tout X:

$$\Pr(\Lambda = X | \mathcal{H}) = \sum_{i=1}^{l} \Pr(\mathcal{H}_i | \mathcal{H}) \Pr(\Lambda = X | \mathcal{H}_i)$$

$$\frac{\Pr(\Lambda = X | \mathcal{H})}{\Pr(\Lambda = X)} = \sum_{i=1}^{l} \Pr(\mathcal{H}_i | \mathcal{H}) \frac{\Pr(\Lambda = X | \mathcal{H}_i)}{\Pr(\Lambda = X)}$$

Lorsque l'on extrait une matrice X_i de X, un certain nombre de bits sont laissés à part parce que n'entrant dans ancun mot de code entier. Nous supposerons que ces bits sont indépendants et de probabilité $1/2$ et l'on a dans ce cas pour tout i:

$$\frac{\Pr(\Lambda = X | \mathcal{H}_i)}{\Pr(\Lambda = X)} = \frac{\Pr(\Lambda_i = X_i | \mathcal{H}_i)}{\Pr(\Lambda_i = X_i)}$$

Et l'on sait (Corollaire 4) que $\frac{\Pr(\Lambda_i = X_i | \mathcal{H}_i)}{\Pr(\Lambda_i = X_i)} = (1 + z^{n_i})^{N_i} \left(\frac{1 - z^{n_i}}{1 + z^{n_i}}\right)^{\mathrm{wt}(\bar{1}.X_i^t)}$ [1].

Ce qui donne finalement:

$$\frac{\Pr(\Lambda = X | \mathcal{H})}{\Pr(\Lambda = X)} = \sum_{i=1}^{l} \Pr(\mathcal{H}_i | \mathcal{H}) (1 + z^{n_i})^{N_i} \left(\frac{1 \Leftrightarrow z^{n_i}}{1 + z^{n_i}}\right)^{\mathrm{wt}(\bar{1}.X_i^t)}$$

Et l'algorithme décidera que $\mathcal{H}$ est vraie si cette dernière expression est supérieure à un seuil S paramétrable.

Pour trouver ensuite (en cas de décision positive) laquelle des hypothèses $\mathcal{H}_i$ est la plus probable, il suffit bien-sûr de déterminer lequel des termes de la somme apporte la plus forte contribution.

Qu'en est-il maintenant des probabilités de fausse alarme et de détection ? Si $\Omega(S) = \left\{(\omega_1 \dots \omega_l) \in \times_{i=1}^{l} \{0 \dots N_i\} / \sum_{i=1}^{l} \Pr(\mathcal{H}_i | \mathcal{H}) (1 + z^{n_i})^{N_i} \left(\frac{1 - z^{n_i}}{1 + z^{n_i}}\right)^{\omega_i} > S\right\}$, on a alors:

$$P_{fa}(S) = \sum_{(\omega_1 \dots \omega_l) \in \Omega(S)} \Pr(\forall i \in \{1 \dots l\} \quad \mathrm{wt}(\bar{1}.X_i^t) = \omega_i)$$

$$P_{det}(S) = \sum_{(\omega_1 \dots \omega_l) \in \Omega(S)} \Pr(\forall i \in \{1 \dots l\} \quad \mathrm{wt}(\bar{1}.X_i^t) = \omega_i | \mathcal{H})$$

1. Le vecteur $\bar{1}$ est de longueur variable, et donc la notation incohérente. J'ai préféré cet abus de notation ponctuel et guère problématique à une notation rigoureuse mais surchargée.

Il est possible d'approximer ces probabilités grâce à des hypothèses plus ou moins abusives d'indépendance statistique entre les éléments constitutifs des évènements à dénombrer. Je propose par exemple l'approximation ci-dessous:

$$\begin{aligned}
P_{fa}(S) &\approx \sum_{(\omega_1 \ldots \omega_l) \in \Omega(S)} \prod_{i=1}^{l} \Pr(\mathrm{wt}(\bar{1}.X_i^t) = \omega_i) \\
&\approx \sum_{(\omega_1 \ldots \omega_l) \in \Omega(S)} \prod_{i=1}^{l} \binom{N_i}{\omega_i} / 2^{N_i} \\
P_{det}(S) &\approx \sum_{j=1}^{l} \Pr(\mathcal{H}_j \,|\, \mathcal{H}) \sum_{(\omega_1 \ldots \omega_l) \in \Omega(S)} \Pr(\mathrm{wt}(\bar{1}.X_j^t) = \omega_j | \, \mathcal{H}_j) \prod_{1 \le i \le l, i \neq j} \Pr(\mathrm{wt}(\bar{1}.X_i^t) = \omega_i) \\
&\approx \sum_{j=1}^{l} \Pr(\mathcal{H}_j \,|\, \mathcal{H}) \sum_{(\omega_1 \ldots \omega_l) \in \Omega(S)} (1 \Leftrightarrow z^{n_j})^{\omega_j} \, (1 + z^{n_j})^{N_j - \omega_j} \prod_{i=1}^{l} \binom{N_i}{\omega_i} / 2^{N_i}
\end{aligned}$$

Mais l'ensemble $\Omega(S)$ est de taille exponentielle par rapport à l, le nombre d'hypothèses, et les sommations seront donc difficilement réalisables. Il sera possible d'encadrer les quantités ci-dessus par d'autres quantités obtenues en ajoutant ou en enlevant certains termes de la somme de manière à obtenir des sommes factorisables, et en corrigeant éventuellement les résultats obtenus (la manière dont on s'y prendra dépendra évidemment du nombre l et de l'effort que l'on est prêt à consacrer au calcul du seuil S), mais observons déjà ce qui suit:

Soit $F'(X) = \log_2 \left(\sum_{i=1}^{l} \Pr(\mathcal{H}_i \,|\, \mathcal{H}) \, (1 + z^{n_i})^{N_i} \left(\frac{1 - z^{n_i}}{1 + z^{n_i}} \right)^{\omega_i} \right)$, le logarithme en base 2 du critère. Pour tout i, on a:

$$F'(X) \geq \log_2(\Pr(\mathcal{H}_i \,|\, \mathcal{H})) + N_i \log_2 (1 + z^{n_i}) + \omega_i \log_2 \left(\frac{1 \Leftrightarrow z^{n_i}}{1 + z^{n_i}} \right)$$

On en déduit que quel que soit i

$$\begin{aligned}
E_i' &\stackrel{def}{=} \mathrm{Esp}(F'(\Lambda) | \, \mathcal{H}_i) \\
&\geq \log_2(\Pr(\mathcal{H}_i \,|\, \mathcal{H})) + N_i \log_2 (1 + z^{n_i}) + \mathrm{Esp}(\omega_i | \, \mathcal{H}_i) \log_2 \left(\frac{1 \Leftrightarrow z^{n_i}}{1 + z^{n_i}} \right) \\
&= \log_2(\Pr(\mathcal{H}_i \,|\, \mathcal{H})) + N_i \log_2 (1 + z^{n_i}) + N_i \frac{1 \Leftrightarrow z^{n_i}}{2} \log_2 \left(\frac{1 \Leftrightarrow z^{n_i}}{1 + z^{n_i}} \right) \\
&= \log_2(\Pr(\mathcal{H}_i \,|\, \mathcal{H})) + N_i \left(1 \Leftrightarrow H_2 \left(\frac{1 \Leftrightarrow z^{n_i}}{2} \right) \right)
\end{aligned}$$

Ces formules sont intéressantes car, si S est le seuil choisi, la détection de l'hypothèse $\mathcal{H}_i$ n'est réaliste que si $E_i' > \log_2(S)$, ce dont on peut facilement se rendre compte. On peut alors considérer l'ensemble $\{\mathcal{H}_i; i \in \{1 \ldots l\} / E_i' > \log_2(S)\}$ des hypothèses que l'on sera en mesure de détecter, et se dispenser des calculs relatifs aux autres hypothèses. Ce faisant on diminue la probabilité de fausse alarme, et l'on réévaluera éventuellement à la baisse le seuil S optimal.

3.3.2 Cas souple

On appelle $L = (a_1 \dots a_m)$ le train de nombres réels intercepté. Et pour chaque hypothèse $\mathcal{H}_i$, appelons n_i la longueur de code, $s_i < n_i$ le décalage (nombre de bits à supprimer en début de train pour se synchroniser selon l'hypothèse), N_i le nombre de mots de code correspondant, L_i la $n_i \times N_i$-matrice extraite de L en fonction de ces valeurs et Λ_i la variable aléatoire correspondante.

De même que précédemment, on montre aisément que:

$$\frac{\rho(\Lambda = L|\,\mathcal{H})}{\rho(\Lambda = L)} = \sum_{i=1}^{l} \Pr(\mathcal{H}_i\,|\,\mathcal{H}) \frac{\rho(\Lambda_i = L_i|\,\mathcal{H}_i)}{\rho(\Lambda_i = L_i)}$$

Et en invoquant le corollaire 8, on obtient:

$$\frac{\rho(\Lambda_N = L|\,\mathcal{H})}{\rho(\Lambda = L)} = \sum_{i=1}^{l} \Pr(\mathcal{H}_i\,|\,\mathcal{H}) \prod_{j=1}^{N_i} \left(1 + \prod_{k=1}^{n_i} \tanh\left(\frac{a_{s_i+n_i(j-1)+k}}{A}\right)\right)$$

Et l'algorithme décidera que $\mathcal{H}$ est vraie si cette dernière expression est supérieure à un seuil S paramétrable.

Pour trouver ensuite (en cas de décision positive) laquelle des hypothèses $\mathcal{H}_i$ est la plus probable, il suffit bien-sûr de déterminer lequel des termes de la somme apporte la plus forte contribution.

Quant à la maîtrise des probabilités de fausse alarme et de détection en fonction de S, elle semble difficile et la méthode semi-empirique me paraît cette fois encore la mieux indiquée. On peut toutefois faire les observations suivantes:

Soit $F'(L) = \log_2\left(\sum_{i=1}^{l} \Pr(\mathcal{H}_i\,|\,\mathcal{H}) \prod_{j=1}^{N_i} \left(1 + \prod_{k=1}^{n_i} \tanh\left(\frac{a_{s_i+n_i(j-1)+k}}{A}\right)\right)\right)$, le logarithme en base 2 du critère. Pour tout i, on a:

$$F'(L) \geq \log_2(\Pr(\mathcal{H}_i\,|\,\mathcal{H})) + \sum_{j=1}^{N_i} \log\left(1 + \prod_{k=1}^{n_i} \tanh\left(\frac{a_{s_i+n_i(j-1)+k}}{A}\right)\right)$$

On en déduit que quel que soit i

$$\begin{aligned} E'_i &\stackrel{def}{=} \mathrm{Esp}(F'(\Lambda)|\,\mathcal{H}_i) \\ &\geq \log_2(\Pr(\mathcal{H}_i\,|\,\mathcal{H})) + N_i\,\mathrm{Esp}\left(\log\left(1 + \prod_{k=1}^{n_i} \tanh\left(\frac{\alpha_k}{A}\right)\right)|\,\mathcal{H}_i\right) \end{aligned}$$

L'espérance de $\log\left(1 + \prod_{k=1}^{n_i} \tanh\left(\frac{\alpha_k}{A}\right)\right)$ sachant $\mathcal{H}_i$ ne dépend que du rapport signal à bruit $\mathcal{E}^2/N_0$ et de la longueur n_i. On pourra donc calculer des abaques dont on pourra déduire E'_i par interpolation.

Si S est le seuil choisi, la détection de l'hypothèse $\mathcal{H}_i$ n'est réaliste que si $E'_i > \log_2(S)$. On pourra alors considérer l'ensemble $\{\mathcal{H}_i; i \in \{1...l\}/E'_i > \log_2(S)\}$ des hypothèses que l'on sera en mesure de détecter, et se dispenser des calculs relatifs aux autres hypothèses. Ce faisant on diminue la probabilité de fausse alarme, et l'on réévaluera éventuellement à la baisse le seuil S optimal.

Conclusion

Nous avons pu juger de l'utilité des différentes considérations du chapitre précédent.

Faire de la détection et de la reconnaissance optimale en utilisant ou non l'information souple, dans le cas d'un code de parité est à portée de calcul, il n'y a donc aucune raison de s'en priver et nous pouvons considérer que ce problème est parfaitement résolu...

On peut, par cette étude, se rendre compte de la longueur maximale $n_{\max}(\tau)$, au delà de laquelle, pour un taux d'erreur binaire τ donné, la détection ne sera pas possible.

Nous nous sommes servis de la relation de parité que doivent vérifier les mots de code. Dans le cas général, les hypothétique mots de code doivent également vérifier des relations de parité qui sont de poids supérieur à la distance dual. On peut, connaissant la longueur n et la dimension k de l'hypothétique code, approximer celle-ci par $nH_2^{-1}(R)$. Lorsque cette quantité est plus grande que la longueur $n_{\max}$ définie ci-dessus, donc lorsque $n > \frac{n_{\max}(\tau)}{H_2^{-1}(R)}$, il ne sera donc pas possible de détecter le code.

Chapitre 4

Cas général: détection optimale et reconnaissance sous contrainte de temps

La vérité scientifique n'arrive d'ordinaire au grand nombre que lorsque elle a cessé d'être vraie.
Jean Rostand (Pensées d'un biologiste)

Mon ami, la vérité vraie est toujours invraisemblable, le savez-vous? Pour rendre la vérité plus vraisemblable, il faut absolument y mêler du mensonge.
Fédor Mikhaïl Dostoievski (Les Démons)

[...] je ne puis croire qu'à de l'invraisemblable. Car le vraisemblable ne serait, selon toute vraisemblance, qu'un produit humain.
Jean Guitton (Mon testament philosophique)

C'est entre ce qui est le plus semblable que l'apparence fait les plus beaux mensonges ; car c'est pardessus le plus petit abîme qu'il est le plus difficile de tendre un pont.
Friedrich Nietsche (Ainsi parlait Zarathoustra)

Introduction

Dans le chapitre précédent, la détection était basée sur l'analyse d'une relation de parité unique et connue, le mot dual correspondant était $\bar{1}$: le mot composé uniquement de 1.

Cette fois, le code à détecter est inconnu. Si le train binaire est effectivement issu d'un codeur linéaire binaire et si le détecteur a fait la bonne hypothèse quant à la synchronisation et la longueur, il faudra non seulement effectuer le même type d'analyse que précédemment sur les relations de parité vérifiées par les mots de code, mais aussi et surtout débusquer ces relations de parité que l'on ne connaît pas d'avance.

La probabilité de détection sera donc d'autant plus élevée que ces relations de parité sont nombreuses, c'est à dire que la dimension du code est faible. La reconnaissance quant à elle sera basée sur une reconstruction d'une matrice de parité du code grâce justement aux relations de parité retrouvées; elle sera donc au contraire d'autant plus difficile que la dimension du code de parité est élevée, et celle du code faible.

Pour « débusquer » une relation de parité h, nous utiliserons le fait que hX^t doit être de poids significativement inférieur à $N/2$. On produira donc grâce à un générateur de mots de poids faibles (section 4.1.2) un mot hX^t de faible poids.

La sortie de ce générateur constitue alors l'observation dont on dispose. Nous verrons alors comment appliquer la théorie de la détection et de la reconnaissance optimales sous contrainte de temps à ce système.

Pour générer des mots de poids faibles, on choisira un algorithme non déterministe de manière à pouvoir le relancer de nombreuses fois et obtenir des résultats différents et si possible indépendants. Ainsi, si la probabilité de détection liée à la génération d'un seul mot est trop faible, il sera possible de l'augmenter en demandant à l'algorithme de générer non pas un mais m mots et de décider que $\mathcal{H}$ est vraie si au moins x ($1 \leq x < m$) de ceux-là vérifient le critère.

Nous avons vu en conclusion du chapitre précédent que la longueur de code détectable (ou reconnaissable) était majorée, nous supposerons la longueur maximale est telle qu'il soit possible de décoder de manière quasi-optimal un mot de cette longueur relativement à n'importe quel code (de cette longueur), en temps raisonnable.

4.1 Générateur de candidats

4.1.1 Moments d'ordre supérieur

Le problème de la détection de code a été abordé en 1996, dans le cadre d'une thèse [Pla96], dans le cas de codes convolutifs, sous un angle différent de celui-ci, moins algébrique, et adapté surtout au petite longueur de code.

Il s'agit d'une étude des moments du train binaire observé. Les concepts classiques de spectre d'autocorrélation ou de densité spectrale de puissance par exemple, qui sont sans doute déjà présents à l'esprit du lecteur familier de traitement du signal, y sont étudiés dans le détail.

Planquette montre que si les bits du train binaire avant encodage valaient 0 ou 1 avec la même probabilité 1/2 (ce qui constitue l'une de nos hypothèse de

travail), alors le train binaire ne possède ni corrélation spectrale ni spectre de raies; et que dans ce cas, seule l'étude des moments d'ordres supérieurs pourra éventuellement permettre de détecter le code.

Cette étude des moments d'ordres supérieurs est précisément équivalente, pour les codes en bloc, à l'étude des poids des $hX^t; h \in \mathbf{F}_2^n$ et au critère du premier ordre que nous allons définir dans la suite. En revanche les critères d'ordres supérieurs (à ne pas confondre avec les *moments* d'ordres supérieurs qui constitue la base du critère du *premier ordre*) que nous étudierons également n'ont pas d'équivalents et s'avèrent bien mieux adaptés aux codes en bloc et aux grandes longueurs.

4.1.2 Génération de moments de poids faible

Dans cette section, nous étudions un algorithme générateur de mots de poids faibles parmi les éléments d'un code linéaire binaire.

J'attire ici l'attention du lecteur sur le fait que nous cherchons des mots de poids faible parmi les combinaisons linéaires des colonnes de X, c'est à dire dans un code linéaire binaire de longueur N et de dimension n, généré par la transposée de la matrice X, nous appellerons ces mots les moments. Ce code n'a évidemment rien de commun avec le code que nous essayons de détecter mais la confusion est si courante que ce rappel ne sera peut-être pas superflu.

La distribution des poids dans un code est en général difficile à calculer exactement et dans les applications pratiques, on se contente d'une règle empirique et approximative: « Si A_i est le nombre de mots de poids i dans un $[N, n]\Leftrightarrow$code binaire, les nombres $\frac{A_i}{2^n}$ sont distribués suivant une loi binômiale de paramètres $\left(\frac{1}{2}, N\right)$. » Pour mesurer la différence entre cette distribution approximative et la distribution réelle, on pourra se référer à [Sol93]; mais dans le cas qui nous intéresse la loi binômiale nous offre un modèle parfaitement satisfaisant.

Définition 13 *Pour tout couple (A, B) d'entiers, nous appelons loi binômiale sur l'intervalle $\{A \dots B\}$, la loi:*

$$\begin{array}{rcl} \{A \dots B\} & \rightarrow & [0,1] \\ W & \mapsto & \frac{\binom{B-A}{W-A}}{2^{B-A}} \end{array}$$

J'appelle « générateur de mots » un algorithme qui, étant donnée une matrice génératrice d'un code linéaire binaire, donne un mot quelconque (aléatoire) élément de ce code. Les poids des mots ainsi générés obéissent à une loi binômiale sur l'intervalle allant de 0 à la longueur du code.

Un générateur de mots de poids faible est un générateur de mots tel que la distribution des poids obtenus est décalée vers les poids plus faibles (la métaphore du « filtre passe-bas » en donne une bonne approche).

La recherche d'un mot de poids minimal dans un code est un problème réputé difficile (NP-dur). Notre problème est de générer des mots non pas de poids minimal mais de poids faible. Ces deux problèmes sont liés de manière évidente

mais avant tout parce que les algorithmes les plus performants recherchant un mot de poids minimal s'appuient sur un générateur de mots de poids faible qui doit être aussi rapide et aussi « filtrant » que possible (ces deux caractéristiques étant bien-sûr antagonistes, un compromis optimal doit être trouvé).

C'est pour cette raison que l'étude des algorithmes de recherche de mots de poids minimal dans un code (comme en particulier la lecture des travaux d'Anne Canteaut et de Florent Chabaud [CC98]) m'a fourni tous les éléments pour construire un bon générateur de moments de poids faible.

Le générateur retenu utilise une diagonalisation selon un ensemble d'information choisi aléatoirement et calcule l'ensemble des mots de poids au plus p dans un code poinçonné de support contenant cet ensemble d'information.

On se réfèrera à la partie « Décodage », et aux sections 2.1.4, p. 43, et 2.3.2, p. 54, pour une étude de l'algorithme plus précise (implémentation, coûts...) que les paragraphes qui suivent ou pour une justification de certains résultats que j'y énonce.

Je rappelle que l'on a affaire à un $[N, n]\Leftrightarrow$code. L'algorihme choisit (aléatoirement) un ordre sur les N positions, puis diagonalise selon cet ordre (ou plutôt met sous la forme dite « échelon ») la matrice X^t.

Les n lignes de la matrice obtenue fournissent déjà des moments de poids faible puisque l'on impose à $n \Leftrightarrow 1$ positions de valoir 0, à une seule de valoir 1 et que les $N \Leftrightarrow n$ autres positions ont une chance sur deux de valoir 0 ou 1 (on obtient donc pour le poids de ces moments une loi binômiale sur l'intervalle $\{1 \dots N \Leftrightarrow n+1\}$). Mais l'on va les filtrer un peu plus avant en utilisant un code poinçonné de longueur s $(n < s < N)$.

L'algorithme choisit donc $s \Leftrightarrow n$ positions au hasard en dehors de la diagonale. Soit alors X_s le code engendré par la matrice X^t en la poinçonnant en $N \Leftrightarrow s$ positions de manière à ce que le support restant contienne l'ensemble d'information et les $s \Leftrightarrow n$ positions choisies, et soit H_s une $(s \Leftrightarrow n) \times s$-matrice de parité de ce code.

Il sépare alors le support en deux parties de même taille $s/2$ puis génère et stocke dans deux tableaux (un tableau pour chacune des deux parties) les demi-moments m (de support inclus dans la partie en cours de traitement) de poids au plus $p/2$ ainsi que leur « syndrôme partiel » $\sigma = H_s m^t$.

Cela étant fait, il parcourt alors les tableaux à la recherche des syndrômes partiels σ tels qu'au moins deux demi-moments m et m' (un pour chaque partie) admettent ce syndrôme partiel. On a alors, pour tous les couples (m, m') trouvés, $H_s(m + m')^t = 0$, ce qui signifie que $m + m'$ appartient au code poinçonné X_s.

On réencode alors les moments $m + m'$ grâce à la matrice X^t diagonalisée, les moments obtenus constituent des moments de poids faible. En effet, ils sont de poids au plus p sur le support de longueur s et de poids en moyenne $\frac{N-s}{2}$ sur son complémentaire. Leur poids est distribué selon une loi binômiale sur l'intervalle $\{p...N \Leftrightarrow s + p\}$.

L'utilisation du code poinçonné sert d'une part à filtrer davantage les moments (pour que la distribution binômiale sur $\{p...N \Leftrightarrow s + p\}$ soit plus intéressante que celle sur $\{1 \dots N \Leftrightarrow n + 1\}$ obtenue par simple diagonalisation, il faut que $p < 1 + \frac{s-n}{2}$), et d'autre part à en obtenir un plus grand nombre à chaque

itération (diminuant le coût moyen par moment généré).

Si , $(s,p) = \sum_{i=0}^{p/2} \binom{s/2}{i}$ est le nombre de « demi-moments » générés sur chaque moitié de support, le nombre moyen de moments complets ainsi générés est $\frac{\Gamma(s,p)^2}{2^{s-n}}$, et le coût moyen de la génération de ces moments, en dehors de la diagonalisation, est:

$$2, (s,p)\frac{p}{2}(s \Leftrightarrow n) + \frac{, (s,p)^2}{2^{s-n}} p(N \Leftrightarrow s)$$

Quant à la diagonalisation (la matrice H_s se déduisant très simplement de la matrice X^t diagonalisée, son calcul ne coûte pour ainsi dire rien), elle coûte en moyenne (cf. Décodage 2.2.4, p. 50) $\frac{n^2(N-n)^2}{2N}$. Si on impose aux ensembles d'information successivement choisis de ne différer entre eux que par une position, alors la diagonalisation ne coûte que $\frac{n(N-n)}{2}$.

Le coût moyen par moment généré est donc:

$$p(N \Leftrightarrow s) + \frac{p(s \Leftrightarrow n)2^{s-n}}{2, (s,p)} + \frac{n^2(N \Leftrightarrow n)^2 2^{s-n}}{2N, (s,p)^2} \tag{4.1}$$

Nous verrons sur les exemples du chapitre 4.3 comment calculer les paramètres s et p optimaux.

4.2 Critères

Supposons donc que l'on ait un algorithme B, prenant en entrée la matrice X, utilisant le générateur de moments de poids faible, et générant une sortie $w \in W = \mathrm{Im}(B)$.

Une partie des mots h et des moments hX^t qui auront été générés pendant l'execution de l'algorithme B figurera sans doute, sous une forme ou sous une autre, dans la sortie de celui-ci. Sans perte de généralités, on peut donc considérer que W est de la forme $\left(\mathbf{F}_2^n \times \mathbf{F}_2^N\right)^r \times W'$, c.a.d. qu'il existe $r \geq 1$ tel qu'une partie de la sortie de B consiste en un $r \Leftrightarrow$*uplet* de couples $(h_1, h_1X^t)...(h_r, h_rX^t)$, le reste étant un élément d'un certain ensemble W'.

La détection requièrera une estimation de:

$$\frac{\Pr(B(\Lambda_N) = w | \mathcal{H})}{\Pr(B(\Lambda_N) = w)}$$

La reconnaissance de longueur et de synchronisation cherchera quant à elle:

$$\mathrm{argmax}_{i=1...l}(\Pr(\mathcal{H}_i \,|\, \mathcal{H}) \Pr(B_i(\Lambda_N) = w_i | \,\mathcal{H}_i))$$

(où $\mathcal{H}_1 ... \mathcal{H}_l$ sont les différentes hypothèses de longueur et de synchronisation, $B_1...B_l$ les algorithmes «B» adaptés à ces hypothèses, et $w_1...w_l$ leurs sorties)

L'étude de ces deux quantités demande certes la connaissance précise de l'algorithme B, mais de part la stratégie adoptée et la forme de $w \in W$, on

peut s'attendre à ce qu'elles soient reliées à des quantités du type $\Pr(\Lambda_N = X|\{h_1...h_r\} \subset C^\perp)$.

Par ailleurs la reconnaissance de code doit trouver $h_1...h_{n-k}$ indépendants qui maximisent l'estimation de $\Pr(\Lambda_N = X|\{h_1...h_{n-k}\} \subset C^\perp)$ qui est également du type mentionné.

Cette section fait précisément l'étude de ces quantités, indépendamment de tout algorithme de détection ou de reconnaissance, lesquels feront l'objet d'autres sections qui se rapporteront à celles-ci.

4.2.1 Notations et remarques

Distinguons donc l'hypothèse faible $\mathcal{H}_1$ selon laquelle la matrice était de rang non plein avant transmission et addition de l'erreur, et l'hypothèse forte $\mathcal{H}(h)$, définie pour tout mot $h \in \mathbf{F}_2^n$, selon laquelle les lignes de la matrice vérifiaient toutes, avant transmission, la relation de parité définie par h (ce qui est equivalent à $h \in C^\perp$).

Nous noterons $h^\perp$ l'ensemble (hyperplan) des mots de $\mathbf{F}_2^n$ vérifiant cette relation.

De même, nous définissons l'hypothèse faible $\mathcal{H}_r$ selon laquelle la matrice était de rang au plus $n \Leftrightarrow r$, et pour tout $r \Leftrightarrow$uplet $(h_1 \dots h_r)$ de mots linéairement indépendants, l'hypothèse forte $\mathcal{H}(h_1 \dots h_r)$ selon laquelle les lignes de la matrice vérifiaient toutes, avant transmission, chacune des relations de parité définies par $h_1 \dots h_r$ (ce qui signifie en particulier que l'hypothétique code est de dimension au plus $n \Leftrightarrow r$).

Nous noterons $\langle h_1 \dots h_r \rangle^\perp$ l'ensemble des mots de $\mathbf{F}_2^n$ vérifiant ces relations, et $\langle h_1 \dots h_r \rangle$ l'ensemble des combinaisons linéaires de $h_1 \dots h_r$.

On a évidemment l'équivalence pour tout $r > 0$:

$$\mathcal{H}_r \Leftrightarrow \exists h_1 \dots h_r \in \mathbf{F}_2^n \quad \mathcal{H}(h_1 \dots h_r)$$

Remarque 6 *Notons que quel que soit le r-uplet $h_1...h_r$, on a bien-sûr:*

$$\frac{\Pr(\Lambda_N = X|\,\mathcal{H})}{\Pr(\Lambda_N = X)} > \Pr(\mathcal{H}(h_1 \dots h_r)|\,\mathcal{H}) \frac{\Pr(\Lambda_N = X|\,\mathcal{H}(h_1 \dots h_r))}{\Pr(\Lambda_N = X)}$$

Ci-dessous, nous étudions $\Pr(\mathcal{H}(h_1 \dots h_r)|\,\mathcal{H})$, nous allons voir que cette probabilité est très petite (inférieure à 2^{-rk}). Mais nous verrons également dans la suite que le ratio $\frac{\Pr(\Lambda_N = X|\,\mathcal{H}(h_1...h_r))}{\Pr(\Lambda_N = X)}$ est quant à lui supérieur à $2^{\alpha r N}$ pour un certain $\alpha > 0$ et peut prendre le dessus pour $N > \frac{k}{\alpha}$.

On pourrait démontrer que si $\mathcal{H}$ est vraie et qu'un code de dimension k est utilisé pour transmettre un train binaire infini, si $\mathcal{H}(h_1 \dots h_{n-k})$ est vraie et si pour tout N, X_N est la matrice des N premiers mots reçus, alors:

$$\lim_{N \to \infty} \frac{\Pr(\Lambda_N = X_N|\,\mathcal{H})}{\Pr(\mathcal{H}(h_1 \dots h_{n-k})|\,\mathcal{H}) \Pr(\Lambda_N = X_N|\,\mathcal{H}(h_1 \dots h_{n-k}))} = 1$$

Essayons dès maintenant d'évaluer $\Pr(\mathcal{H}(h_1 \dots h_r)|\,\mathcal{H})$. Il s'agit de la probabilité pour que le code dual de l'hypothétique code à détecter contienne les vecteurs $h_1 ... h_r$. Nous supposerons que tous les codes de longueur et de dimension données sont équiprobables.

Si $D_{k,n}$ désigne toujours l'hypothèse selon laquelle la dimension du code vaut k et sa longueur n, nous avons:

$$\Pr(\mathcal{H}(h_1 \dots h_r)|\,\mathcal{H}) = \sum_{k=1}^{n-r} \Pr(D_{k,n}|\,\mathcal{H}) \Pr(\mathcal{H}(h_1 \dots h_r)|D_{k,n})$$

On se réfèrera à la section 1.2.2, p. 101, pour les valeurs de $\Pr(D_{k,n}|\,\mathcal{H})$.

Proposition 18

$$\Pr(\mathcal{H}(h_1 \dots h_r)|D_{k,n}) = \prod_{i=0}^{r-1} \frac{2^{n-k-i} \Leftrightarrow 1}{2^{n-i} \Leftrightarrow 1} (< 2^{-rk})$$

Démonstration: Par équiprobabilité $\Pr(\mathcal{H}(h_1 \dots h_r)|D_{k,n})$, qui est la probabilité pour que $C^\perp$ contienne $h_1 ... h_r$ sachant que C est de dimension k, est la probabilité pour qu'un sous espace vectoriel de $\mathbf{F}_2^n$ de dimension $n \Leftrightarrow k$ contienne r mots indépendants donnés et est égale au rapport du nombre de tels sous-espace vectoriels au nombre total de sous espace vectoriels de $\mathbf{F}_2^n$ de dimension $n \Leftrightarrow k$.

Tout sous-espace vectoriel de dimension $n \Leftrightarrow k$ se représente de manière unique par une $(n \Leftrightarrow k) \times n$-matrice si on impose à celle ci d'être par exemple sous forme échelon (systématique le « plus à gauche possible », sur l'ensemble d'information minimal pour l'ordre lexicographique). Appelons G l'ensemble de ces matrices que l'on ne peut systématiser davantage vers la gauche selon cet ordre. G est donc en bijection avec l'ensemble des sous-espaces vectoriels de dimension $n \Leftrightarrow k$.

Par ailleurs, soit R l'ensemble des $(n \Leftrightarrow k) \times (n \Leftrightarrow k)$-matrice de rang $n \Leftrightarrow k$ et N l'ensemble des $(n \Leftrightarrow k) \times n$-matrice de rang $n \Leftrightarrow k$. Le produit d'un élément de R par un élément de G donne un élément de N, et inversement la diagonalisation d'un élément de N donne un élément de G et un élément de R. Une bijection existe donc entre $G \times R$ et N.

On pourra vérifier dans [LN83], p. 455 que le nombre de $u \times v$-matrice binaire $(u \leq v)$ de rang u est $\prod_{i=0}^{u-1} (2^v \Leftrightarrow 2^i)$.

On en déduit que le nombre de sous-espaces vectoriels de dimension $n \Leftrightarrow k$, égal à $\frac{|N|}{|R|}$, vaut:

$$\prod_{i=0}^{n-k-1} \frac{2^n \Leftrightarrow 2^i}{2^{n-k} \Leftrightarrow 2^i}$$

Remarque 7 *Par involution de la dualité, ce nombre est évidemment égal à celui des sous-espaces vectoriels de dimension k, c.a.d. $\prod_{i=0}^{k-1} \frac{2^n - 2^i}{2^k - 2^i}$ (même s'il n'est pas trivial a priori que ces deux formules soient égales).*

De même, tout sous-espace vectoriel de dimension $n \Leftrightarrow k$ contenant $r < n \Leftrightarrow k$ mots $h_1 ... h_r$ indépendants donnés se représente de manière unique par le produit du sous-espace vectoriel engendré par $h_1 ... h_r$ et d'un sous-espace vectoriel de dimension $n \Leftrightarrow k \Leftrightarrow r$ si on impose aux éléments de celui-ci d'être nuls sur un ensemble d'information donné du premier.

En répétant l'argumentation de la première partie de cette démonstration, on trouve que le nombre de ces sous-espaces vectoriels est le rapport du nombre de $(n \Leftrightarrow k \Leftrightarrow r) \times (n \Leftrightarrow r)$-matrices de rang $n \Leftrightarrow k \Leftrightarrow r$ par le nombre de $(n \Leftrightarrow k \Leftrightarrow r) \times (n \Leftrightarrow k \Leftrightarrow r)$-matrices de rang $n \Leftrightarrow k \Leftrightarrow r$, soit:

$$\prod_{i=0}^{n-k-r-1} \frac{2^{n-r} \Leftrightarrow 2^i}{2^{n-k-r} \Leftrightarrow 2^i}$$

Le rapport de cette deuxième quantité par la première, qui est la probabilité recherchée, vaut:

$$\frac{\prod_{i=0}^{n-k-r-1} \frac{2^{n-r}-2^i}{2^{n-k-r}-2^i}}{\prod_{i=0}^{r-1} \frac{2^n-2^i}{2^{n-k}-2^i} \prod_{i=r}^{n-k-1} \frac{2^{n-r}-2^{i-r}}{2^{n-k-r}-2^{i-r}}} = \prod_{i=0}^{r-1} \frac{2^{n-k} \Leftrightarrow 2^i}{2^n \Leftrightarrow 2^i}$$

Et en divisant haut et bas par 2^i, on parvient à la formule de la proposition. □

4.2.2 Cas dur

Proposition 19 *Pour tout mot de l'espace ambiant x, tout taux d'erreur binaire τ et tout mot de l'espace ambiant h, on a (avec $z = 1 \Leftrightarrow 2\tau$):*

$$\Pr(\lambda = x) = 2^{-n}$$

$$\Pr(\lambda = x | \mathcal{H}(h)) = 2^{-n} \left(1 + (\Leftrightarrow 1)^{\langle h,x \rangle} z^{\mathrm{wt}(h)}\right)$$

Démonstration: En l'absence d'hypothèse, les 2^n mots de l'espace ambiant sont equiprobables. On a donc $\Pr(\lambda = x) = 2^{-n}$. Sous l'hypothèse $\mathcal{H}(h)$ en revanche, seuls les 2^{n-1} mots c de $h^\perp$ (tels que $\langle h, c \rangle = 0$) ont pu être transmis. On a donc:

$$\begin{aligned} \Pr(\lambda = x | \mathcal{H}(h)) &= \sum_{c \in h^\perp} \Pr(\gamma = c) \Pr(\delta = x + c) \\ &= 2^{1-n} \sum_{c \in h^\perp} \Pr(\delta = x + c) \end{aligned}$$

Mais $(\exists c \in h^\perp \quad \delta = x + c) \Leftrightarrow (\langle h, \delta \rangle = \langle h, x \rangle)$, donc $\sum_{c \in h^\perp} \Pr(\delta = x + c) = \Pr(\langle h, \delta \rangle = \langle h, x \rangle)$. Supposons $\langle h, x \rangle = 1$, $\Pr(\langle h, \delta \rangle = 1)$ est la probablité pour qu'il y ait un nombre impair d'erreur sur le support de h, soit (cf. Lemme 3) $\frac{1-z^{\mathrm{wt}(h)}}{2}$. De même $\Pr(\langle h, \delta \rangle = 0) = \frac{1+z^{\mathrm{wt}(h)}}{2}$.

Par conséquent $\Pr(\lambda = x | \mathcal{H}(h)) = 2^{-n} \left(1 + (\Leftrightarrow 1)^{\langle h,x \rangle} z^{\mathrm{wt}(h)}\right)$. □

Corollaire 9 *Pour toute matrice X, tout taux d'erreur binaire τ et tout mot de l'espace ambiant h, on a (avec $z = 1 \Leftrightarrow 2\tau$):*

$$\Pr(\Lambda_N = X) = 2^{-nN}$$

$$\Pr(\Lambda_N = X \,|\, \mathcal{H}(h)) = 2^{-nN} \left(1 + z^{\mathrm{wt}(h)}\right)^N \left(\frac{1 \Leftrightarrow z^{\mathrm{wt}(h)}}{1 + z^{\mathrm{wt}(h)}}\right)^{\mathrm{wt}(hX^t)}$$

On s'attend donc à ce que les critères du premier ordre soient liés de manière monotone à cette quantité. Nous étudions un algorithme pour lequel nous définissons un tel critère en section 4.3. Or, on a bien-sûr:

$$\forall S \in \mathbf{R} \quad \forall w \in \{1 \dots n\} \quad \exists S(w) \in \{0 \dots N+1\} \quad \forall W \in \{0 \dots N\}$$

$$(1 + z^w)^N \left(\frac{1 \Leftrightarrow z^w}{1 + z^w}\right)^W > S \Longleftrightarrow W < S(w)$$

Un algorithme (du premier ordre) commencera donc par stocker les valeurs adéquates pour $S(w)$ ($w = 1 \dots n$), puis cherchera des mots h tels que $\mathrm{wt}(hX^t) < S(\mathrm{wt}(h))$.

Remarque 8 *L'espérance de $\langle h, \lambda \rangle$ est $\frac{1}{2}$ en général, et $\frac{1-z^{\mathrm{wt}(h)}}{2}$ si $\mathcal{H}(h)$ est vraie. Les lignes λ de Λ_N étant identiques et indépendantes, la loi faible des grands nombres nous indiquent que si $\frac{1-z^n}{2} < \alpha < \frac{1}{2}$ alors:*

$$\lim_{N \to \infty} \Pr(\mathrm{wt}(h\Lambda_N^t) < \alpha N) = 0, \qquad \lim_{N \to \infty} \Pr(\mathrm{wt}(h\Lambda_N^t) < \alpha N \,|\, \mathcal{H}(h)) = 1$$

Ceci implique que, si l'on suppose que l'on n'est pas limité en temps de calcul ni en nombre de mots reçus (ce qui n'est bien sûr pas le cas mais la remarque est intéressante), on pourra obtenir quelque certitude quant à l'appartenance de certains mots h au code dual de l'hypothétique code. En particulier, si l'on trouve que $\mathcal{H}(h)$ est vraie avec quasi-certitude, alors bien-sûr $\mathcal{H}$ est vraie avec au moins la même certitude.

Si l'on désire davantage de précision, on pourra se référer à la démonstration du corollaire 6, p. 124 pour montrer que:

Pour tout $h \in \mathbf{F}_2^n$, soient $\beta = \frac{1-z^{\mathrm{wt}(h)}}{2}$, α vérifiant $\beta < \alpha < \frac{1}{2}$, $K_1 = 2^{H_2(\alpha)-1}$ et $K_2 = (\frac{\beta}{\alpha})^\alpha (\frac{1-\beta}{1-\alpha})^{1-\alpha}$, on a:

$$0 < K_1 < 1, \qquad 0 < K_2 < 1,$$

et pour tout N:

$$\begin{aligned} \Pr(\mathrm{wt}(h\Lambda_N^t) < \alpha N) &\leq K_1^N \\ \Pr(\mathrm{wt}(h\Lambda_N^t) < \alpha N \,|\, \mathcal{H}(h)) &\geq 1 \Leftrightarrow \frac{\alpha(1 \Leftrightarrow \beta)}{\alpha \Leftrightarrow \beta} K_2^N \end{aligned}$$

Un générateur de moments de poids faible génèrera par conséquent une part de moments correspondant à des mots du dual d'autant plus importante que son pouvoir filtrant est élevé.

Sur la figure 4.1, les couples de poids $(\mathrm{wt}(h), \mathrm{wt}(hX^t))$ généré par un algorithme du type décrit en section 4.1.2 sont tracés. La figure illustre la séparation qui existe entre des couples $(\mathrm{wt}(h), \mathrm{wt}(hX^t))$ correspondant à des mots h quelconques ou des mots h appartenant au dual.

Les niveaux de gris sont proportionnels au nombre d'occurrences obtenues pour un couple donné. Lorsque le taux d'erreurs binaires augmente, on est obligé d'augmenter N pour que les deux catégories de couples soient séparées. Mais lorsque N augmente, le pouvoir filtrant du générateur de moments de poids faible diminue, et le nombre de couples appartenant à la catégorie des mots du dual devient négligeable devant le nombre des mots quelconques (pour la troisième figure, il m'a fallu réhausser les niveaux de gris de la catégorie des mots du dual).

La matrice X que l'on reçoit, si elle était de rang non plein avant addition de l'erreur, semble donc se distinguer d'une matrice aléatoire par le fait que le sous-espace-vectoriel de $\mathbf{F}_2^N$ engendré par X^t contient des mots de poids anormalement faible.

Si l'on parvient à séparer ainsi les deux catégories de couples, un critère du premier ordre permettra de distinguer les deux catégories de mots et sera suffisant pour la détection et la reconnaissance. Si au contraire, le taux d'erreur binaire et le nombre N de mots reçus sont tels que la séparation est ambigüe, il faudra un critère plus discriminant.

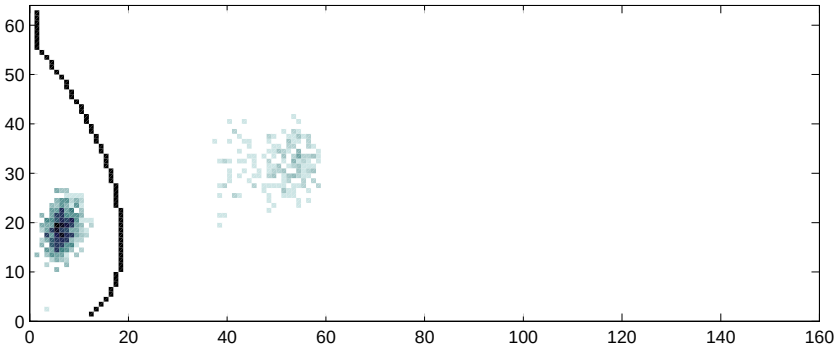
Si l'on fait l'hypothèse plus forte que la matrice X était de rang au plus $n \Leftrightarrow r$ $(r > 1)$ avant addition de l'erreur, elle doit vérifier la propriété suivante: le sous-espace-vectoriel de $\mathbf{F}_2^N$ engendré par X^t contient lui-même un sous-espace-vectoriel (de dimension r) composé de mots de poids anormalement faibles. Un critère d'ordre r consiste à mesurer l'ampleur de cette propriété, qui est bien-sûr beaucoup plus forte que la propriété considérée par un critère du premier ordre.

Il calculera une fonction d'un sous-espace vectoriel $\langle h_1 ... h_r \rangle$ donné. Cette fonction dépend de l'algorithme précis qui est utilisé pour générer ce sous-espace vectoriel, mais sera généralement liée à la probabilité $\Pr(\Lambda_N = X | \mathcal{H}(h_1 ... h_r))$ que nous calculons dans cette section.

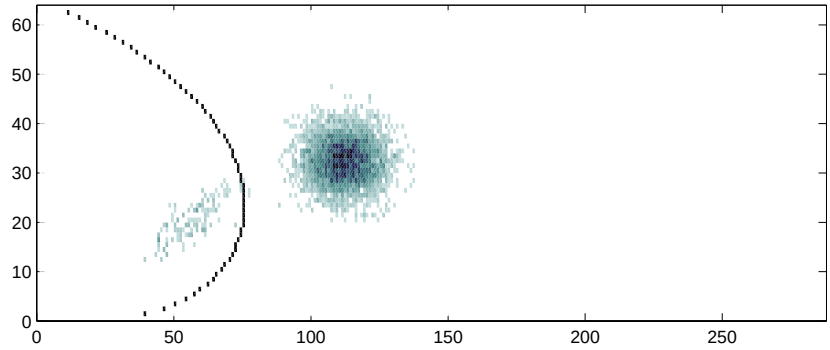
Proposition 20 *Pour tout mot de l'espace ambiant x, tout taux d'erreur binaire τ et toute $r \times n$ matrice $H = \begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$, on a (avec $z = 1 \Leftrightarrow 2\tau$):*

$$\Pr(\lambda = x | \mathcal{H}(h_1 \dots h_r)) = 2^{-n} \sum_{B \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle B, xH^t \rangle} z^{\mathrm{wt}(BH)}$$

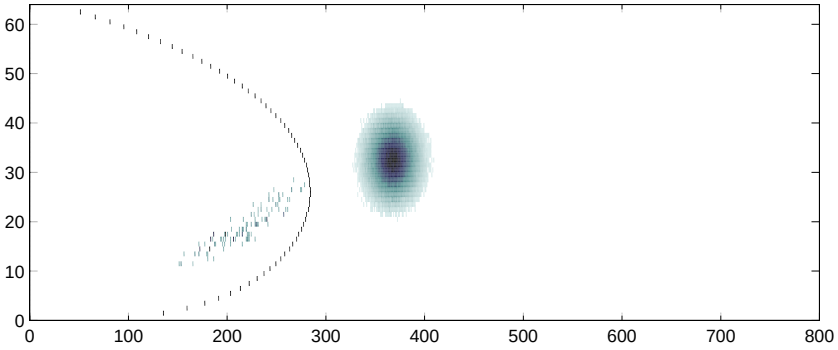
Démonstration: La démonstration la plus concise de ce résultat se base sur une identité de MacWilliams à propos des « énumérateurs exacts » (cf. [MS77], Théorème 14, p. 147). Je l'énonce ci-après, adaptée au cas binaire.



(a) taux d'erreur binaire: 0.5/64, N: 160



(b) taux d'erreur binaire: 1/64, N: 288



(c) taux d'erreur binaire: 1.5/64, N: 800

FIG. 4.1: *Constellations des couples de poids* $(\mathrm{wt}(h), \mathrm{wt}(hX^t))$ *obtenu par l'algorithme du premier ordre pour un [64,34]-code*

Lemme 4 *Soit $t = ((t_{10}, t_{11}), (t_{20}, t_{21}) \dots (t_{n0}, t_{n1}))$ un $n\Leftrightarrow$uplet de couple de variables. On définit l'énumérateur exact associé à t d'un code C sur $\mathbf{F}_2^n$ comme étant:*

$$\mathcal{E}_C^{(t)} = \sum_{u \in C} t_{1u_1} t_{2u_2} \dots t_{nu_n}.$$

Le théorème de MacWilliams pour l'énumérateur exact affirme que:

$$\mathcal{E}_C^{(t)} = \frac{\mathcal{E}_{C^\perp}^{(t')}}{|C^\perp|}$$

$$\textit{avec } t'_{i0} = t_{i0} + t_{i1} \textit{ et } t'_{i1} = t_{i0} \Leftrightarrow t_{i1} \ (i = 1 \dots n).$$

D'après l'hypothèse $\mathcal{H}(h_1 \dots h_r)$, seuls les mots de $\langle h_1 \dots h_r \rangle^\perp$ ont pu être transmis. On a donc:

$$\begin{aligned}
\Pr(\lambda = x | \mathcal{H}(h_1 \dots h_r)) &= \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \Pr(\gamma = c) \Pr(\delta = x + c) \\
&= 2^{r-n} \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \Pr(\delta = x + c) \\
&= 2^{r-n} \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \tau^{\mathrm{wt}(x+c)} (1 \Leftrightarrow \tau)^{n - \mathrm{wt}(x+c)} \\
&= 2^{r-n} \mathcal{E}_{\langle h_1 \dots h_r \rangle^\perp}^{(t)}
\end{aligned}$$

avec $t_{i0} = \begin{cases} 1 \Leftrightarrow \tau & \text{si } x_i = 0, \\ \tau & \text{si } x_i = 1. \end{cases}$ et $t_{i1} = \begin{cases} \tau & \text{si } x_i = 0, \\ 1 \Leftrightarrow \tau & \text{si } x_i = 1. \end{cases}$.

Afin d'appliquer le lemme, nous définissons donc $t'_{i0} = t_{i0} + t_{i1} = 1$ et $t'_{i1} = t_{i0} \Leftrightarrow t_{i1} = \begin{cases} z & \text{si } x_i = 0, \\ \Leftrightarrow z & \text{si } x_i = 1. \end{cases}$. Et si $\langle h_1 \dots h_r \rangle$ représente le code engendré par $h_1 \dots h_r$, on a:

$$\begin{aligned}
\Pr(\lambda = x | \mathcal{H}(h_1 \dots h_r)) &= 2^{-n} \mathcal{E}_{\langle h_1 \dots h_r \rangle}^{(t')} \\
&= 2^{-n} \sum_{c \in \langle h_1 \dots h_r \rangle} (\Leftrightarrow 1)^{\langle c, x \rangle} z^{\mathrm{wt}(c)} \\
&= 2^{-n} \sum_{B \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle BH, x \rangle} z^{\mathrm{wt}(BH)}
\end{aligned}$$

Et $\langle BH, x \rangle = BHx^t = \langle B, xH^t \rangle$. La proposition en découle. □

Corollaire 10 *Si $d = \min_{c \in \langle h_1 \dots h_r \rangle^\perp} \mathrm{wt}(x + c)$, alors:*

$$\sum_{B \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle B, xH^t \rangle} z^{\mathrm{wt}(BH)} > 2^r \tau^d (1 \Leftrightarrow \tau)^{n-d}$$

Démonstration: Cela provient du fait, mentionné dans la démonstration précédente, que:

$$\begin{aligned}\Pr(\lambda = x | \, \mathcal{H}(h_1 \dots h_r)) &= 2^{-n} \sum_{B \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle B, xH^t \rangle} z^{\mathrm{wt}(BH)} \\ &= 2^{r-n} \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \tau^{\mathrm{wt}(x+c)} (1 \Leftrightarrow \tau)^{n - \mathrm{wt}(x+c)}\end{aligned}$$

□

Corollaire 11 *Pour toute matrice* $X = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$, *tout taux d'erreur binaire* τ *et toute* $r \times n$ *matrice* $H = \begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$, *on a (avec* $z = 1 \Leftrightarrow 2\tau$*):*

$$\Pr(\Lambda_N = X | \, \mathcal{H}(h_1 \dots h_r)) = 2^{-nN} \prod_{i=1}^{N} \sum_{B \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle B, x_i H^t \rangle} z^{\mathrm{wt}(BH)}$$

Un critére d'ordre r consistera souvent à comparer une fonction croissante de cette derniére quantité à un seuil et à répondre positivement en cas de supériorité. C'est pourquoi, il pourra parfois suffire d'utiliser directement cette quantité sans faire l'étude des probabilités réelles sous-jacentes à la manière dont on génère des candidats. En général, cependant, cette étude est indispensable au bon dimensionnement des seuils et autres paramètres des algorithmes mis au point.

4.2.3 Cas souple

Nous nous plaçons encore dans le cadre d'une modulation antipodal d'énergie e^2 et d'un canal à bruit additif, blanc, gaussien de variance $N0$.

Je rappelle que:

- β représente le bit transmis et vaut 0 ou 1 avec une probabilité 1/2.
- γ représente le mot de code transmis En l'absence d'hypothèse, il y a 2^n possibilités equiprobables pour γ: tous les mots binaires de longueur n. Sous l'hypothèse $\mathcal{H}(h_1 \dots h_r)$ en revanche, γ doit être élément du sous-espace-vectoriel $\langle h_1 \dots h_r \rangle^\perp$ composé des mots vérifiant les relations de parité définies par $h_1 \dots h_r$, ce qui donne 2^{n-r} possibilités equiprobables.
- ϵ représente le bruit. Pour le canal sans mémoire, à bruit blanc, gaussien et additif de densité monolatérale $N0/2$, nous avons:

$$\forall e \in \mathbf{R} \qquad \rho(\epsilon = e) = \frac{1}{\sqrt{\pi N0}} \exp\left(\Leftrightarrow \frac{e^2}{N0}\right)$$

– δ représente l'erreur de transmission, produit combinatoire de n variables aléatoires ϵ. Sa densité de probabilité est:

$$\forall d \in \mathbf{R}^n \qquad \rho(\delta = d) = (\pi N0)^{-n/2} e^{-\frac{\sum_{i=1}^n d_i^2}{N0}}$$

– α représente le signal « reçu », nombre réel, somme du signal transmis ($\pm\mathcal{E}$) et du bruit. Sa densité de probabilité est définie par celles de β et ϵ. C'est le réel qui est mesuré par le démodulateur.

– Enfin λ représente le mot « reçu », n⇔uplet de nombres réels, somme du mot transmis modulé et de l'erreur de transmission, et Λ_N est le produit combinatoire de N variables aléatoires λ, représenté par une $N \times n$⇔matrice réelle. Sa densité de probabilité est définie par celle de γ et δ.

L'observation dont on dispose est une réalisation L de la variable aléatoire Λ_N.

Rappelons aussi que pour alléger les notations, nous définissons la quantité $A = \frac{N0}{2\,\mathcal{E}}$. Ainsi, nous avons:

$$\Pr(\beta = 0 | \alpha = a) = \frac{e^{a/A}}{2\cosh(a/A)} \qquad \Pr(\beta = 1 | \alpha = a) = \frac{e^{-a/A}}{2\cosh(a/A)}$$

Proposition 21 *Pour tout n⇔uplets $(a_1 \dots a_n)$ de nombres réels, tout mot h et tout r⇔uplet $(h_1 \dots h_r)$ on a, en notant $K = \frac{\exp\left(-\frac{\sum_{i=1}^n a_i^2 + n\,\mathcal{E}^2}{N0}\right)}{(\pi N0)^{n/2}}$:*

$$\begin{aligned}
\rho(\lambda = a_1 \dots a_n) &= K \prod_{i=1}^n \cosh\left(\frac{a_i}{A}\right) \\
\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h)) &= K \prod_{i=1}^n \cosh\left(\frac{a_i}{A}\right) \left(1 + \prod_{i \in \mathrm{supp}(h)} \tanh\left(\frac{a_i}{A}\right)\right) \\
\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r)) &= K \prod_{i=1}^n \cosh\left(\frac{a_i}{A}\right) \sum_{B \in \mathbf{F}_2^r} \prod_{i \in \mathrm{supp}(BH)} \tanh\left(\frac{a_i}{A}\right)
\end{aligned}$$

où $\mathrm{supp}(x)$ *représente le support d'un mot x, et où par convention un produit sur l'ensemble vide vaut* 1.

Démonstration: Pour $b \in \mathbf{F}_2$ et $a \in \mathbf{R}$, la densité de probabilité de mesurer a si b est émis vaut $\rho(\alpha = a | \beta = b) = \frac{1}{\sqrt{\pi N0}} e^{-\frac{(a-(-1)^b\,\mathcal{E})^2}{N0}} = \frac{\exp\left(-\frac{a^2+\mathcal{E}^2}{N0}\right)}{\sqrt{\pi N0}} e^{(-1)^b a/A}$.

Dans le premier cas, les a_i sont indépendants, donc $\rho(\lambda = a_1 \dots a_n) = \prod_{i=1}^n \rho(\alpha = a_i)$.

Mais $\rho(\alpha = a) = \frac{1}{2}(\rho(a|0) + \rho(a|1)) = \frac{\exp\left(-\frac{a^2+\mathcal{E}^2}{N0}\right)}{\sqrt{\pi N0}} \cosh\left(\frac{a}{A}\right)$, d'où la première formule.

La seconde formule correspond précisément à la troisième dans le cas où $r = 1$. Démontrons donc celle-ci dans le cas général: on a:

$$\begin{aligned}
&\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r)) \\
&\quad = \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \Pr(\gamma = c | \, \mathcal{H}(h_1 \dots h_r)) \rho(\lambda = a_1 \dots a_n | \gamma = c) \\
&\quad = 2^{r-n} \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \prod_{i=1}^{n} \rho(\alpha = a_i | \beta = c_i) \\
&\quad = \frac{K}{2^{n-r}} \sum_{c \in \langle h_1 \dots h_r \rangle^\perp} \exp\left(\sum_{i=1}^{n} (\Leftrightarrow 1)^{c_i} a_i / A \right)
\end{aligned}$$

Nous allons utiliser le lemme 4: Notons pour $i = 1 \dots n$: $t_{i0} = e^{\frac{a_i}{A}}$ et $t_{i1} = e^{\frac{-a_i}{A}}$, nous venons de voir que:

$$\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r)) = \frac{K}{2^{n-r}} \mathcal{E}^{(t)}_{\langle h_1 \dots h_r \rangle^\perp}$$

Par application du lemme, nous avons donc:

$$\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r)) = \frac{K}{2^n} \mathcal{E}^{(t')}_{\langle h_1 \dots h_r \rangle}$$

(avec $t'_{i0} = t_{i0} + t_{i1} = 2\cosh\left(\frac{a_i}{A}\right)$) et $t'_{i1} = t_{i0} \Leftrightarrow t_{i1} = 2\sinh\left(\frac{a_i}{A}\right)$))
Soit:

$$\begin{aligned}
&\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r)) \\
&\qquad = K \prod_{i=1}^{n} \cosh\left(\frac{a_i}{A}\right) \sum_{h \in \langle h_1 \dots h_r \rangle} \prod_{i \in \mathrm{supp}(h)} \tanh\left(\frac{a_i}{A}\right)
\end{aligned}$$

D'où les deux dernières formules □

Corollaire 12

$$\forall (a_1 \dots a_n) \in \mathbf{R}^n \qquad \forall h_1 \dots h_r \in \mathbf{F}_2^n$$

$$\frac{\rho(\lambda = a_1 \dots a_n | \, \mathcal{H}(h_1 \dots h_r))}{\rho(\lambda = a_1 \dots a_n)} = \sum_{h \in \langle h_1 \dots h_r \rangle} \prod_{i \in \mathrm{supp}(h)} \tanh\left(\frac{a_i}{A}\right)$$

Corollaire 13 *Pour toute $N \times n$-matrice réelle L dont les coefficients sont notés $a_{i,j}$ $(1 \leq i \leq N, 1 \leq j \leq n)$ et tout r⇔uplet $(h_1 \dots h_r)$, on a:*

$$\frac{\rho(\Lambda_N = L | \, \mathcal{H}(h_1 \dots h_r))}{\rho(\Lambda_N = L)} = \prod_{i=1}^{N} \sum_{h \in \langle h_1 \dots h_r \rangle} \prod_{j \in \mathrm{supp}(h)} \tanh\left(\frac{a_{i,j}}{A}\right)$$

Cette quantité sera donc de première importance dans la mise au point d'un critère de détection d'ordre r utilisant l'information souple.

4.2.4 Coûts de calcul des critères

H représente la matrice $\begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$.

Nous considèrerons que le coût des additions, multiplications et exponentiations de nombres entiers ou réels est unitaire, que le calcul d'une tangente hyperbolique coûte c_t, que le calcul du poids d'un vecteur binaire de longueur l coûte $c_w l$, que l'addition de deux tels vecteurs coûte $c_a l$ et leur produit scalaire $c_p l$.

Et pour des raisons qui apparaîtront plus clairement lorsque nous considèrerons l'implémentation effective d'un algorithme de détection, nous considérons que le calcul de $\langle h, x\rangle$, hX^t, Hx^t et HX^t a été pris en charge par l'algorithme de recherche de mots de poids faibles (nous ne tenons donc pas compte du coût de leur calcul pour celui des critères).

Critère	Valeur	Coût de calcul	Dominance
$\frac{\Pr(\lambda=x \mid \mathcal{H}(h))}{\Pr(\lambda=x)}$	$1 + (\Leftrightarrow 1)^{\langle h,x\rangle} z^{\mathrm{wt}(h)}$	$c_{\mathrm{w}} n + 3$	$c_{\mathrm{w}} n$
$\frac{\Pr(\Lambda_N=X \mid \mathcal{H}(h))}{\Pr(\Lambda_N=X)}$	$(1+z^{\mathrm{wt}(h)})^N \left(\frac{1-z^{\mathrm{wt}(h)}}{1+z^{\mathrm{wt}(h)}}\right)^{\mathrm{wt}(hX^t)}$	$c_{\mathrm{w}}(n+N)+6$	$c_{\mathrm{w}}(n+N)$
$\frac{\Pr(\lambda=x \mid \mathcal{H}(H))}{\Pr(\lambda=x)}$	$\sum_{B\in\mathbf{F}_2^r} (-1)^{\langle B, xH^t\rangle} z^{\mathrm{wt}(BH)}$	$2^r(c_{\mathrm{p}} r + (c_{\mathrm{a}} r/2 + c_{\mathrm{w}})n + 3)$	$\frac{c_{\mathrm{a}}}{2} 2^r rn$
$\frac{\Pr(\Lambda_N=X \mid \mathcal{H}(H))}{\Pr(\Lambda_N=X)}$	$\prod_{i=1}^{N} \sum_{B\in\mathbf{F}_2^r} (-1)^{\langle B, x_i H^t\rangle} z^{\mathrm{wt}(BH)}$	$N_H 2^r\left(c_{\mathrm{p}} r + \left(\frac{c_{\mathrm{a}}}{2} r + c_{\mathrm{w}}\right)n + 3\right) + N_H + Nr$	$\frac{c_{\mathrm{a}}}{2} N_H 2^r rn$
$\frac{\rho(\Lambda_N=L \mid \mathcal{H}(H))}{\rho(\Lambda_N=L)}$	$\prod_{i=1}^{N} \sum_{B\in\mathbf{F}_2^r} \prod_{j\in\mathrm{supp}(BH)} \tanh\left(\frac{a_{i,j}}{A}\right)$	$N 2^r n\left(\frac{c_{\mathrm{a}}}{2} r + \frac{c_{\mathrm{t}}}{2} + 1\right) + N 2^r + Nr$	$\frac{c_{\mathrm{a}}}{2} N 2^r rn$

où N_H est le nombre de colonnes différentes dans la matrice HX^t ($N_H \leq \min(2^r, N)$).

Et l'on voit que le coût de calcul d'un critère est plus qu'exponentiel en son ordre, si bien que l'on sera fortement limité par rapport à celui-ci.

Cela dit, pour comparer deux hypothèses $\mathcal{H}(h_1 ... h_r)$ et $\mathcal{H}(h'_1 ... h'_r)$ de même ordre r (avec $\langle h_1 ... h_r\rangle \neq \langle h'_1 ... h'_r\rangle$) lorsque r est trop élevé pour un calcul «exact» des probabilités, on pourra générer un certain nombre K de mots de poids faibles h et h' dans $\langle h_1 ... h_r\rangle$ et $\langle h'_1 ... h'_r\rangle$ respectivement et comparer:

$$\sum_{i=1}^{N} \sum_{h} (\Leftrightarrow 1)^{\langle h, x_i\rangle} z^{\mathrm{wt}(h)} \qquad \text{à} \qquad \sum_{i=1}^{N} \sum_{h'} (\Leftrightarrow 1)^{\langle h', x_i\rangle} z^{\mathrm{wt}(h')}$$

Ou bien, si l'on dispose d'information souple

$$\sum_{i=1}^{N} \sum_{h} \prod_{j\in\mathrm{supp}(h)} \tanh\left(\frac{a_{i,j}}{A}\right) \qquad \text{à} \qquad \sum_{i=1}^{N} \sum_{h'} \prod_{j\in\mathrm{supp}(h')} \tanh\left(\frac{a_{i,j}}{A}\right)$$

Nous y reveindrons plus tard lors de l'étude des algorithmes de reconnaissance.

4.3 Détection

4.3.1 Exemple d'algorithme

Le but de cette section est de se faire une idée des ordres de grandeur et des dépendances des probabilités de détection et de fausse alarme, afin de mieux se rendre compte des facteurs qui limiteront l'efficacité d'un algorithme, et pour mieux dimensionner ses paramètres.

Elle offre en outre un exemple d'application de la proposition 13, et de mise au point d'algorithme de détection. Cet exemple n'est cependant pas le meilleur algorithme de détection que ce manuscrit puisse inspirer car il n'utilise qu'un critère du premier ordre.

Cas « dur »

Soit B l'algorithme de recherche de moments de poids faible décrit en section 4.1.2, avec ses paramètres s et p. Je rappelle brièvement son principe en passant sur les détails algorithmiques et en posant quelques notations adaptées à cette partie:

- Choisir aléatoirement s mots reçus parmi les N; et considérer la matrice X_s composée de ces mots.
- Calculer des mots h tels que les moments hX_s^t de la matrice X_s soient de poids au plus p.
- Calculer les moments complets $y = hX^t$.

Les poids des moments produits par cet algorithme sont distribués statistiquement selon une loi binômiale sur l'intervalle $\{p \dots N \Leftrightarrow s + p\}$. Nous supposerons que les moments produits successivement sont statistiquement indépendants.

Pour chaque moment ainsi généré, l'algorithme a imposé à p positions de valoir 1 et à $s \Leftrightarrow p$ autres positions de valoir 0. Pour faire l'étude du critère, de $\beta(S) = P_{det}(S,1,1)$ et de $\gamma(S) = P_{fa}(S,1,1)$, nous allons considérer un seul moment. Nous pouvons donc supposer, sans perte de généralités, que ces positions sont fixées. Par exemple, si $x_1 ... x_N$ sont les lignes de X, on impose $\langle h, x_i \rangle = 1$ pour $i = 1...p$, et $\langle h, x_i \rangle = 0$ pour $i = p+1...s$.

Adoptons les notations suivantes:

$$X^t = (\overbrace{X_1^t}^{p} \mid \overbrace{X_0^t}^{s-p} \mid \overbrace{X_\bullet^t}^{N-s}), \qquad X_{10}^t = (X_1^t | X_0^t),$$

$$y_1 = (\overbrace{1 \dots 1}^{p}), \qquad y_0 = (\overbrace{0 \dots 0}^{s-p}), \qquad y_{10} = (y_1 | y_0)$$

Et pour tout vecteur y de longueur N, on notera $y_\bullet$ sa restriction aux $N \Leftrightarrow s$ dernières positions.

B génère donc le seul mot h tel que $hX_{10}^t = y_{10}$. Il calcule par ailleurs le moment $y = hX^t = (y_{10}, y_\bullet)$, dont seule la partie $y_\bullet$ est inconnue. On considèrera donc que la sortie de l'algorithme B est le couple $(h, y_\bullet)$.

Soit $\mathcal{H}$ l'hypothèse selon laquelle les mots reçus (les lignes de X) sont des éléments d'un même code linéaire binaire de dimension k ayant traversé le canal binaire symétrique de probabilité de transition τ. X est l'observation d'une variable aléatoire Λ dont la loi dépend du fait que cette hypothèse est vraie ou fausse. On lui applique l'algorithme B ci-dessus, et l'on observe alors sa sortie $(h, y_\bullet)$.

On définit alors, pour tout couple $(h, y_\bullet)$ de $F_2^n \times F_2^{N-s}$:

$$P_d(h, y_\bullet) = \Pr(B(\Lambda) = (h, y_\bullet) | \mathcal{H})$$

$$P_f(h, y_\bullet) = \Pr(B(\Lambda) = (h, y_\bullet))$$

$$\Omega(S) = \left\{ (h, y_\bullet) \in F_2^n \times F_2^{N-s} / \frac{P_d(h, y_\bullet)}{P_f(h, y_\bullet)} > S \right\}$$

La détection aura lieu si le mot h généré par B est tel que $(h, hX_\bullet^t) \in \Omega(S)$. Les probabilités de détection et de fausse alarme mesurent alors la probabilité pour que ce soit le cas selon que l'hypothèse de codage est juste ou fausse. Elles valent:

$$P_{det}(S, 1, 1) = \sum_{(h, y_\bullet) \in \Omega(S)} P_d(h, y_\bullet)$$

$$P_{fa}(S, 1, 1) = \sum_{(h, y_\bullet) \in \Omega(S)} P_f(h, y_\bullet)$$

Pour adapter nos notations aux variables aléatoires on définit donc:

$$\Lambda^t = (\overbrace{\Lambda_1^t}^{p} \mid \overbrace{\Lambda_0^t}^{s-p} \mid \overbrace{\Lambda_\bullet^t}^{N-s}), \qquad \Lambda_{10}^t = (\Lambda_1^t | \Lambda_0^t)$$

Proposition 22 *Soit $y = (y_{10} | y_\bullet)$. On a:*

$$\Pr(B(\Lambda) = (h, y_\bullet)) = \frac{\Pr(h\Lambda^t = y)}{\sum_{h' \in F_2^{n*}} \Pr(h'\Lambda_{10}^t = y_{10})}$$

(il est loisible de rajouter une condition de type « sachant $\mathcal{H}$ » en queue de ces probabilités)

Démonstration: $B(\Lambda)$ est donc un couple $(h(\Lambda), h(\Lambda)\Lambda_\bullet^t)$, et l'on a:

$$\Pr(B(\Lambda) = (h, y_\bullet)) = \Pr(h(\Lambda) = h, h(\Lambda)\Lambda_\bullet^t = y_\bullet)$$

Or $h(\Lambda)$ est défini comme étant le seul mot de l'espace ambiant tel que $h(\Lambda)\Lambda_{10}^t = y_{10}$. Etant donné une propriété, la probabilité pour que $h(\Lambda)$

vérifie cette propriété peut être vue comme la probabilité pour qu'un variable aléatoire μ représentant un mot quelconque de l'espace ambiant la vérifie, sachant que $\mu\Lambda_{10} = y_{10}$. On a donc:

$$\begin{aligned}\Pr(B(\Lambda) = (h, y_\bullet)) &= \Pr(\mu = h, \mu\Lambda_\bullet^t = y_\bullet | \mu\Lambda_{10}^t = y_{10}) \\ &= \frac{\Pr(\mu = h, \mu\Lambda^t = y)}{\Pr(\mu\Lambda_{10}^t = y_{10})} \\ &= \frac{\Pr(\mu = h)\Pr(h\Lambda^t = y)}{\sum_{h' \in F_2^{n*}} \Pr(\mu = h')\Pr(h'\Lambda_{10}^t = y_{10})}\end{aligned}$$

On déduit la proposition du fait qu'en l'absence de condition, les différentes valeurs de μ sont *a priori* equiprobables □

Proposition 23 *Pour tout h non nul et tout $y_\bullet$, on a:*

$$P_f(h, y_\bullet) = \frac{1}{(2^n \Leftrightarrow 1)2^{N-s}}$$

Démonstration: En l'absence de l'hypothèse $\mathcal{H}$, la matrice Λ est considérée comme totalement aléatoire.

On a donc $\Pr(h\Lambda^t = y) = 2^{-N}$ d'une part, car chacune des N positions de $h\Lambda^t$ vaut 0 ou 1 avec la même probabilité et indépendamment des autres.

Et d'autre part $\sum_{h' \in F_2^{n*}} \Pr(h'\Lambda_{10}^t = y_{10}) = (2^n \Leftrightarrow 1)2^{-s}$ car quel que soit h' non nul, chacune des s positions de $h'\Lambda_{10}^t$ vaut 0 ou 1 avec la même probabilité et indépendamment des autres.

On déduit le résultat du fait que $P_f(h, y_\bullet) = \frac{\Pr(h\Lambda^t = y)}{\sum_{h' \in F_2^{n*}} \Pr(h'\Lambda_{10}^t = y_{10})}$. □

Proposition 24 *Pour tout h non nul et tout $y_\bullet$, on a:*

$$P_d(h, y_\bullet) = \frac{1 \Leftrightarrow 2^{-k} + 2^{-k}\left(1 \Leftrightarrow z^{\mathrm{wt}(h)}\right)^{p+\mathrm{wt}(y_\bullet)}\left(1 + z^{\mathrm{wt}(h)}\right)^{N-\mathrm{wt}(y_\bullet)-p}}{2^{N-s}\left((1 \Leftrightarrow 2^{-k})(2^n \Leftrightarrow 1) + 2^{-k}\sum_{w=1}^{n}\binom{n}{w}(1 \Leftrightarrow z^w)^p(1 + z^w)^{s-p}\right)}$$

Démonstration: Nous supposons donc que $\mathcal{H}$ est vraie et qu'un code de dimension k a été utilisé; à l'intérieur de cette démonstration, nous considérons comme implicite la condition « sachant que $\mathcal{H}$ est vraie » dans l'expression des probabilités.

On sait que $P_d(h, y_\bullet) = \frac{\Pr(h\Lambda^t = y)}{\sum_{h' \in F_2^{n*}} \Pr(h'\Lambda_{10}^t = y_{10})}$.

Bien-sûr, les probabilités qui sont en jeu dépendent fortement du fait que h (ou h') appartient ou non au dual du code. Nous supposerons que tout mot de l'espace ambiant appartient au dual avec une probabilité (*a priori*) égale à 2^{-k}. On a alors:

$$P_d(h, y_\bullet) = \frac{2^{-k}\Pr(h\Lambda^t = y | h \in C^\perp) + (1-2^{-k})\Pr(h\Lambda^t = y | h \notin C^\perp)}{\sum_{h' \in F_2^{n*}}(2^{-k}\Pr(h'\Lambda_{10}^t = y_{10} | h \in C^\perp) + (1-2^{-k})\Pr(h'\Lambda_{10}^t = y_{10} | h \notin C^\perp))}$$

Si h n'appartient pas à $C^\perp$, alors, de même que dans le calcul de P_f, chaque positions de $h\Lambda^t$ vaut 0 ou 1 avec la même probabilité et indépendamment des autres; et de même pour $h'\Lambda^t_{10} = y_{10}$ si $h' \notin C^\perp$.

On a donc:

$$P_d(h, y_\bullet) = \frac{2^{-k} \Pr(h\Lambda^t = y | h \in C^\perp) + \frac{1-2^{-k}}{2^N}}{(2^n \Leftrightarrow 1)\frac{1-2^{-k}}{2^s} + 2^{-k} \sum_{h' \in F_2^{n\star}} \Pr(h'\Lambda^t_{10} = y_{10} | h \in C^\perp)}$$

Par ailleurs si on note λ la VA associée à un mot reçu (une ligne de Λ), on a $\Pr(h\Lambda^t = y | h \in C^\perp) = \prod_{i=1}^N \Pr(h\lambda^t = y_i | h \in C^\perp)$ et de même $\Pr(h'\Lambda^t_{10} = y_{10} | h \in C^\perp) = \Pr(h\lambda^t = 1 | h \in C^\perp)^p \Pr(h\lambda^t = 0 | h \in C^\perp)^{s-p}$.

Or si $h \in C^\perp$, la probabilité pour que $h\lambda^t$ soit égal à 0 est la probabililté pour que l'erreur de transmission subie par λ soit de poids pair sur le support de h. D'après le lemme 3, elle vaut $\frac{1+z^{\mathrm{wt}(h)}}{2}$. La probabilité pour que $h\lambda^t$ soit égal à 1 vaut donc, quant à elle $\frac{1-z^{\mathrm{wt}(h)}}{2}$ (de même pour tout h').

Puisque $\mathrm{wt}(y) = p + \mathrm{wt}(y_\bullet)$, on a donc:

$$\Pr(h\Lambda^t = y | h \in C^\perp) = 2^{-N} \left(1 \Leftrightarrow z^{\mathrm{wt}(h)}\right)^{p+\mathrm{wt}(y_\bullet)} \left(1 + z^{\mathrm{wt}(h)}\right)^{N-\mathrm{wt}(y_\bullet)-p}$$

Et pour tout h' de poids w:

$$\Pr(h'\Lambda^t_{10} = y_{10} | h \in C^\perp) = 2^{-s} (1 \Leftrightarrow z^w)^p (1 + z^w)^{s-p}$$

Et l'on retrouve la formule de la proposition en combinant tous ces résultats.

□

Il est intéressant que cette fonction ne dépende que du poids de h et de $y_\bullet$, car les sommations seront ainsi plus aisées.

Définissons les fonctions:

$$g(w, W) = (1 \Leftrightarrow z^w)^{W+p} (1 + z^w)^{N-W-p}$$

$$f(n, k, s, p) = 1 \Leftrightarrow 2^{-k} + \frac{2^{-k}}{2^n \Leftrightarrow 1} \sum_{w=1}^{n} \binom{n}{w} (1 \Leftrightarrow z^w)^p (1 + z^w)^{s-p}$$

Ainsi, on a:

$$P_d(h, y_\bullet) = \frac{1 + 2^{-k}(g(\mathrm{wt}(h), \mathrm{wt}(y_\bullet)) \Leftrightarrow 1)}{2^{N-s}(2^n \Leftrightarrow 1) f(n, k, s, p)}$$

$$\frac{P_d(h, y_\bullet)}{P_f(h, y_\bullet)} = \frac{1 + 2^{-k}(g(\mathrm{wt}(h), \mathrm{wt}(y_\bullet)) \Leftrightarrow 1)}{f(n, k, s, p)}$$

On définit également l'ensemble suivant, qui présente l'intérêt d'être de taille inférieure à nN:

$$\Omega'(S) = \left\{ (w,W) \in \{1...n\} \times \{0...N \Leftrightarrow s\} / \frac{1 + 2^{-k}(g(w,W) \Leftrightarrow 1)}{f(n,k,s,p)} > S \right\}$$

Et l'on a alors:

$$\Omega(S) = \{(h, y_\bullet) / (\mathrm{wt}(h), \mathrm{wt}(y_\bullet)) \in \Omega'(S)\}$$

Il est alors tout à fait envisageable (si l'on connait k), de calculer:

$$P_{det}(S,1,1) = \sum_{(h,y)\in\Omega(S)} P_d(h, y_\bullet) = \sum_{(w,W)\in\Omega'(S)} \frac{\binom{N-s}{W-p}}{2^{N-s}} \frac{\binom{n}{w}}{2^n \Leftrightarrow 1} \frac{1 + 2^{-k}(g(w,W) \Leftrightarrow 1)}{f(n,k,s,p)}$$

$$P_{fa}(S,1,1) = \sum_{(h,y)\in\Omega(S)} P_f(h, y_\bullet) = \sum_{(w,W)\in\Omega'(S)} \frac{\binom{N-s}{W-p}}{2^{N-s}} \frac{\binom{n}{w}}{2^n \Leftrightarrow 1}$$

Remarque 9 *On a donc* $P_{det}(S,1,1) > S \sum_{(w,W)\in\Omega'(S)} \frac{\binom{N-s}{W-p}}{2^{N-s}} \frac{\binom{n}{w}}{2^n}$. *Les poids* W *sélectionnés dans l'ensemble* $\Omega'(S)$ *sont concentrés vers les poids faibles, loin de* $N \Leftrightarrow s + p$. *Cet ensemble ne varie donc pas si l'on augmante* s. *En revanche, les quantités* $\frac{\binom{N-s}{W-p}}{2^{N-s}}$ *augmente rapidement avec* s, *de même donc, que* $P_{det}(S,1,1)$. *Le critère est donc d'autant plus discriminant que* s *est grand.*

Remarque 10 *Calculer les premiers moments, si* $\mathcal{H}$ *est vraie, de* $g(w,W)$, *ou de* $\log(g(w,W))$, *ou de toute fonction croissante de* $\frac{P_d(B(\Lambda))}{P_f(B(\Lambda))}$, *présente peu d'intérêt ici car il est possible de calculer* $P_{det}(S,1,1)$ *et* $P_{fa}(S,1,1)$ *exactement et à un coût raisonnable. Je laisse cependant les résultats suivants (avec uniquement quelques indications pour leur preuve) pour ceux qui voudraient s'approprier intuitivement des ordres de grandeur:*

Soit la fonction (probabiliste) $F(X) = g(\mathrm{wt}(h), \mathrm{wt}(y_\bullet))$, *avec* $(h, y_\bullet) = B(X)$, *on a, en notant* $l_w^{\pm}$ *la quantité* $\log\left(\frac{1+z^w}{1-z^w}\right)$, l_w *la quantité* $\log(1+z^w)$ *et* $\binom{sp}{kw}$ *la quantité* $2^{-k}(1 \Leftrightarrow z^w)^p (1+z^w)^{s-p}$:

$$\mathrm{Esp}(F(\Lambda) | \mathcal{H}) = \frac{\sum_{w=1}^{n} \left((2^k \Leftrightarrow 1)\binom{sp}{kw} + 2^k \binom{sp}{kw}^2 (1+z^{2w})^{N-s} \right)}{(2^n \Leftrightarrow 1) f(n,k,s,p)}$$

$$\mathrm{Esp}(F(\Lambda)^2 | \mathcal{H}) = \frac{\sum_{w=1}^{n} \left(2^k (2^k \Leftrightarrow 1)\binom{sp}{kw}^2 (1+z^{2w})^{N-s} + 2^{2k}\binom{sp}{kw}^3 (1+3z^{2w})^{N-s} \right)}{(2^n \Leftrightarrow 1) f(n,k,s,p)}$$

$$\mathrm{Esp}(\log(F(\Lambda)) | \mathcal{H}) = \frac{\sum_{w=1}^{n} \left(\left(1 \Leftrightarrow 2^{-k} + \binom{sp}{kw}\right) \left(N l_w \Leftrightarrow \left(p + \frac{N-s}{2}\right) l_w^{\pm} \right) + \binom{sp}{kw} (N \Leftrightarrow s) \frac{z^w}{2} l_w^{\pm} \right)}{(2^n \Leftrightarrow 1) f(n,k,s,p)}$$

$$\mathrm{Esp}(\log(F(\Lambda))^2|\,\mathcal{H}) =$$
$$\left(\sum_{w=1}^{n}\left(\left(1 \Leftrightarrow 2^{-k} + \binom{sp}{kw}\right)\left(\left(Nl_w + \left(p \Leftrightarrow \frac{N \Leftrightarrow s}{2}\right) l_w^{\pm}\right)^2 + (p+s)\left(\frac{l_w^{\pm}}{2}\right)^2\right)\right.\right.$$
$$\left.\left. + \binom{sp}{kw}(N \Leftrightarrow s) z^w l_w^{\pm}\left(Nl_w \Leftrightarrow \left(p + \frac{N \Leftrightarrow s}{2} \Leftrightarrow \frac{N \Leftrightarrow s \Leftrightarrow 1}{4} z^w\right) l_w^{\pm}\right)\right)\right) / (2^n \Leftrightarrow 1) f(n,k,s,p)$$

Démonstration: (Indications) *Quelle que soit la fonction $F'(h, y_\bullet)$ considérée, on a:*

$$\mathrm{Esp}(F'(B(\Lambda))|\,\mathcal{H}) = \sum_{h, y_\bullet} P_d(h, y_\bullet) F'(h, y_\bullet)$$

Le fait que $\sum_{W=0}^{N-s} \frac{\binom{N-s}{W}}{2^{N-s}} x^W y^{N-s-W} = \left(\frac{x+y}{2}\right)^{N-s}$ sera suffisant pour calculer les moments de F.

Pour ceux de $\log(F)$, on considèrera en outre le fait que

$$\sum_{W=0}^{N-s} \frac{\binom{N-s}{W}}{2^{N-s}} W x^W y^{N-s-W} = \frac{(N \Leftrightarrow s)x}{2}\left(\frac{x+y}{2}\right)^{N-s-1}$$

$$\sum_{W=0}^{N-s} \frac{\binom{N-s}{W}}{2^{N-s}} W^2 x^W y^{N-s-W} = \frac{(N \Leftrightarrow s)x}{2}\frac{(N \Leftrightarrow s)x + y}{2}\left(\frac{x+y}{2}\right)^{N-s-2}$$

□

On constatera en principe que ces espérances prévoient que $\frac{P_d(B(\Lambda))}{P_f(B(\Lambda))}$ soit très inférieur à $\frac{1}{P_{fa}^{max}}$, et donc que la probabilité de détection associée à un tel seuil soit très faible. Cependant la fonction B est probabiliste, et les valeurs successives qu'elle prendra, si on la calcule m fois, seront indépendantes. On dimensionnera donc m convenablement pour que la probabilité de détection soit proche de 1.

Par ailleurs, on pourra également observer que $2^{\mathrm{Esp}(\log(F))} < \mathrm{Esp}(F)$ (ce que l'on doit à la concavité du logarithme), mais qu'en revanche $\frac{\mathrm{Esp}(\log(F))}{\sqrt{Var(\log(F))}}$ est nettement plus fort que $\frac{\mathrm{Esp}(F)}{\sqrt{Var(F)}}$ donnant plus de puissance aux inégalités de Chebyshev, donc plus de précision.

On justifie ainsi naturellement la considération de «log-likelyhood-ratios» tels que $\log(F)$.

Paramétrage de l'algorithme

Quid du problème de la dimension?

La fonction $P_d(h, y_\bullet)$ et l'ensemble $\Omega(S)$ définis en section précédente dépendent de k que l'on supposait connu, ce qui en fait n'est pas le cas. Nous allons maintenant les noter $P_d^{(k)}(h, y_\bullet)$ et $\Omega^{(k)}(S)$, afin de pouvoir différencier les quantités calculées selon différentes hypothèses de dimension.

Par ailleurs la fonction $P_f(h, y_\bullet) = \frac{1}{(2^n-1)2^{N-s}}$ ne dépendant ni de h, ni de $y_\bullet$, ni de k, nous la noterons simplement P_f.

La proposition suivante découle directement des définitions de ses termes et ne demande pas de démonstration:

Proposition 25 *Si $\tilde{k}$ est la véritable dimension du code, et si le critère est $\frac{P_d^{(k)} B(X)}{P_f} \geq S$ (k peut être différent de $\tilde{k}$) alors la probabilité de détection est $P_{det}(S,1,1) = \sum_{(h,y_\bullet)\in\Omega^{(k)}(S)} P_d^{(\tilde{k})}(h, y_\bullet)$; et la probabilité de fausse alarme est $P_{fa}(S,1,1) = |\Omega^{(k)}(S)|P_f$ (et ne dépend pas de $\tilde{k}$).*

Proposition 26 *Pour tout $k < n \Leftrightarrow 1$, tout h, tout $y_\bullet$ et tout S, on a:*

$$\left(\exists k' \quad P_d^{(k')}(h, y_\bullet) > P_f\right) \Rightarrow P_d^{(k)}(h, y_\bullet) > P_d^{(k+1)}(h, y_\bullet) > P_f$$

$$S > 1 \Rightarrow \Omega^{(k+1)}(S) \subset \Omega^{(k)}(S)$$

Démonstration: Soient h et $y_\bullet$ donnés. Adoptons les notations suivantes:

$$A = g(\mathrm{wt}(h), \mathrm{wt}(y_\bullet)) \Leftrightarrow 1$$

$$A' = \sum_{w=1}^{n} \frac{\binom{n}{w}}{2^n \Leftrightarrow 1}(1 \Leftrightarrow z^w)^p (1 + z^w)^{s-p} \Leftrightarrow 1$$

On a alors, pour tout k', $P_d^{(k')}(h, y_\bullet) = \frac{1+2^{-k'}A}{1+2^{-k'}A'} P_f$, donc:

$$\left(\exists k' \quad P_d^{(k')}(h, y_\bullet) > P_f\right) \Rightarrow A > A'$$

Or, pour tout k, $A > A' \Rightarrow \frac{1+2^{-k}A}{1+2^{-k}A'} > \frac{1+2^{-k-1}A}{1+2^{-k-1}A'} > 1$. On a donc la première partie de la proposition. La deuxième s'en déduit facilement. En effet, pour tout $S > 1$, $\frac{P_d^{(k+1)}(h,y_\bullet)}{P_f} > S \Rightarrow \frac{P_d^{(k)}(h,y_\bullet)}{P_f} > S$, donc:

$$(h, y_\bullet) \in \Omega^{(k+1)}(S) \Rightarrow (h, y_\bullet) \in \Omega^{(k)}(S)$$

□

Corollaire 14 *Si $\tilde{k}$ est la véritable dimension du code, et si le critère est $\frac{P_d^{(k)} B(X)}{P_f} \geq S$ avec $k \geq \tilde{k}$, alors:*

$$\sum_{(h,y_\bullet)\in\Omega^{(k)}(S)} P_d^{(k)}(h, y_\bullet) \leq P_{det}(S,1,1) \leq \sum_{(h,y_\bullet)\in\Omega^{(\tilde{k})}(S)} P_d^{(\tilde{k})}(h, y_\bullet)$$

$$P_{fa}(S,1,1) = |\Omega^{(k)}(S)|P_f$$

La situation est d'autant plus défavorable que k est élevé. Par sécurité, on supposera donc $k = k_{\max}$ (cf. section 1.2.2, p. 101), et l'on appliquera le critère associé.

Par conséquent, pour m, x et S donné, si l'on calcule $P_{det}(S, m, x)$ et $P_{fa}(S, m, x)$ en tenant compte de l'hypothèse $k = k_{\max}$, on sous-estimera la vraie probabilité de détection (tant mieux) tandis que la probabilité de fausse alarme sera juste.

En faisant l'hypothèse $k = k_{\max}$, la proposition 13 s'applique donc bien à notre algorithme.

Par exemple, pour $x = 1$, l'algorithme de détection pourra présenter une probabilité de détection supérieure à P_{det}^{min} et une probabilité de fausse alarme inférieure à P_{fa}^{max}, si l'on choisit $S = \frac{-\ln(1-P_{det}^{min})}{P_{fa}^{max}}$ (il est nécessaire que $\Omega^{(k_{\max})}(S) \neq \emptyset$), et si l'on génère exactement $m = \frac{-\ln(1-P_{det}^{min})}{P_{det}(S,1,1)}$ moments de poids faible.

On pourrait être tenté d'en générer davantage, mais ce faisant, on augmenterait la probabilité de fausse alarme.

Il est cependant généralement plus intéressant de prendre $x > 1$, car alors, pour un même couple $(P_{det}^{min}, P_{fa}^{max})$, le seuil S sera plus petit, et m pourra donc l'être aussi (m est toutefois sujet à une influence antagoniste: si le seuil S ne change pas, m augmente avec x; il existe donc un x optimal), l'algorithme devra étudier les différentes possibilités en calculs préliminaires.

La complexité de l'algorithme est donc égale à m fois la complexité moyenne pour générer un moment de poids faible (le coût de calcul du critère est ici négligeable). Ces deux multiplicandes dépendent des paramètres s et p. Pour optimiser ces derniers, on minimisera bien-sûr le produit.

Ainsi que mentionné en début de section cependant, le critère utilisé, fonction d'un couple $(h, y_\bullet)$, est moins discriminant que ne le serait un critère d'ordre supérieur à 1. En particulier, si l'on attend x détections simples, il sera nettement plus intéressant de leur appliquer des critères d'ordre compris entre 1 et x.

Avec ces critères, les différentes possibilités commencent à se multiplier en d'innombrables modalités. Le nombre de paramètres à régler devient trop grand. Le choix d'un algorithme pour générer des candidats sera difficilement effectué de manière optimale...

Je fais tout de même quelques suggestions à ce propos dans une prochaine section, mais nous n'étudierons pas le détail des probabilités de détection et de fausse alarme des algorithmes ainsi définis. Pour faire ce travail pour un algorithme particulier, cette section pourra servir de référence.

Les critéres d'ordre quelconque et utilisant éventuellement l'information souple étant plus discriminants, ils permettront à l'algorithme de détection d'utiliser un nombre m plus petit, et éventuellement d'avoir une complexité moindre (attention toutefois au fait que le coût de calcul d'un critère est plus qu'exponentiel en son ordre).

Une solution de bon sens, mais laissant peu d'espoir de quantifier correctement les probabilités de fausse alarme et de détection (on pourra tout au plus les borner grossièrement), consiste à sélectionner une liste de x candidats grâce un critère du premier ordre utilisant un seuil S_1 assez faible; à raffiner cette

sélection grâce à un critère du second ordre appliqué à l'ensemble des paires de candidats avec un seuil S_2 plus élevé; et ainsi de suite jusqu'à un ultime raffinement grâce à un critére d'ordre r appliqué aux $r\Leftrightarrow$uplets de candidats restant, avec un seuil S_r correspondant au ratio $\frac{P_{det}}{P_{fa}}$ souhaité. La probabilité de détection dépend de tous les seuils $S_1...S_r$. Ce que doivent être les rapports de grandeur entre ces seuils est encore peu clair pour moi...

Utilisation de l'information souple dans ce même algorithme

On pourra relire la section 3.1 pour se remémorer les notations et leur signification.

On reçoit donc un ensemble de signaux réels, que l'on dispose dans une matrice $L = (a_{ij})_{i=1...N,j=1...n}$. Si l'on note $\Pr(\beta = b|\alpha = a)$ la probabilité pour que le bit envoyé vaille b si le réel a est mesuré, nous avons, avec $A = \frac{N0}{2\mathcal{E}}$:

$$\Pr(\beta = 0|\alpha = a) = \frac{e^{a/A}}{2\cosh(a/A)} \qquad \Pr(\beta = 1|\alpha = a) = \frac{e^{-a/A}}{2\cosh(a/A)}$$

La matrice binaire X des mots reçus est telle que $X_{ij} = 0$ si et seulement si $a_{ij} > 0$. En effet la probabilité pour que le bit envoyé soit X_{ij} ainsi défini si a_{ij} est mesuré vaut alors $\frac{e^{|a|/A}}{2\cosh(a/A)} > \frac{1}{2}$ (la probabilité pour que X_{ij} soit une mauvaise décision, c.a.d. une erreur, vaut donc $\frac{e^{-|a|/A}}{2\cosh(a/A)} < \frac{1}{2}$).

L'algorithme B générateur de candidats sera le même que pour le cas dur. C'est à dire qu'il cherchera des moments de poids de Hamming faible en travaillant sur la matrice binaire X^t.

Grâce à l'information souple, il sera possible de rendre le critère plus discriminant. Tous les résultats énoncés précédemment sont encore valables, mais l'information souple permet un calcul plus précis de l'une des probabilités mentionnées, ainsi que le montre le lemme suivant.

Lemme 5 *Si le signal $(a_1...a_n)$ est reçu, et que l'on considère que le mot binaire reçu vaut $(x_1...x_n)$ avec $x_j = 0$ si et seulement si $a_j > 0$, alors la probabilité pour que l'erreur (dure) de transmission, induite par cette décision, soit de poids pair sur le support d'un mot h vaut:*

$$\frac{1 + \prod_{j\in \mathrm{supp}(h)} \tanh \frac{|a_j|}{A}}{2}$$

Démonstration: Restreignons-nous au support de h, et supposons $\mathrm{wt}(h) = w$.

Le canal étant sans mémoire, quel que soit le mot e (de longueur w) considéré, la probabilité pour que l'erreur de transmission (sur le support de h) soit égale à e vaut:

$$\begin{aligned}\Pr(\delta = e) &= \prod_{j/e_j=1} \frac{e^{-|a_j|/A}}{2\cosh(a_j/A)} \prod_{j/e_j=0} \frac{e^{|a_j|/A}}{2\cosh(|a_j|/A)} \\ &= \frac{e^{\sum_{j\in \mathrm{supp}(h)} (-1)^{e_j} |a_j|/A}}{2^w \prod_{j\in \mathrm{supp}(h)} \cosh(a_j/A)}\end{aligned}$$

Soit E l'ensemble des mots de support inclus dans le support de h, E_0 le sous-ensemble de E constitué des mots de poids poids pair, et E_1 son complément. La probabilité recherchée vaut $\sum_{e\in E_0}\Pr(\delta=e)$.

Or on a:

$$\prod_{j\in\mathrm{supp}(h)}(e^{|a_j|/A}+e^{-|a_j|/A})=\sum_{e\in E}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}$$
$$=\sum_{e\in E_0}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}+\sum_{e\in E_1}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}$$

$$\prod_{j\in\mathrm{supp}(h)}(e^{|a_j|/A}\Leftrightarrow e^{-|a_j|/A})=\sum_{e\in E}(\Leftrightarrow 1)^{\mathrm{wt}(e)}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}$$
$$=\sum_{e\in E_0}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}\Leftrightarrow\sum_{e\in E_1}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}$$

Donc

$$\prod_{j\in\mathrm{supp}(h)}(e^{|a_j|/A}+e^{-|a_j|/A})+\prod_{j\in\mathrm{supp}(h)}(e^{|a_j|/A}\Leftrightarrow e^{-|a_j|/A})$$
$$=2^w\left(\prod_{j\in\mathrm{supp}(h)}\cosh(|a_j|/A)+\prod_{j\in\mathrm{supp}(h)}\sinh(|a_j|/A)\right)$$
$$=2\sum_{e\in E_0}e^{\sum_{j\in\mathrm{supp}(h)}(-1)^{e_j}|a_j|/A}$$

La probabilité recherchée vaut donc:

$$\frac{\prod_{j\in\mathrm{supp}(h)}\cosh(\frac{|a_j|}{A})+\prod_{j\in\mathrm{supp}(h)}\sinh(\frac{|a_j|}{A})}{2\prod_{j\in\mathrm{supp}(h)}\cosh(\frac{|a_j|}{A})}=\frac{1+\prod_{j\in\mathrm{supp}(h)}\tanh(\frac{|a_j|}{A})}{2}$$

□

Corollaire 15 *La probabilité pour que l'erreur de transmission e vérifie $\langle h,x\rangle=\langle h,e\rangle$ vaut:*

$$\frac{1+\prod_{j\in\mathrm{supp}(h)}\tanh\frac{a_j}{A}}{2}$$

Démonstration: La probabilité recherchée vaut $\frac{1+(-1)^{\langle h,x\rangle}\prod_{j\in\mathrm{supp}(h)}\tanh\frac{|a_j|}{A}}{2}$.

Or, par parité de la tangente:

$$\prod_{j\in\mathrm{supp}(h)} \tanh\frac{a_j}{A} = \prod_{j\in\mathrm{supp}(h)} \mathrm{sgn}\, a_j \tanh\frac{|a_j|}{A}$$
$$= \prod_{j\in\mathrm{supp}(h)} (\Leftrightarrow 1)^{x_j} \tanh\frac{|a_j|}{A} = (\Leftrightarrow 1)^{\langle h,x\rangle} \prod_{j\in\mathrm{supp}(h)} \tanh\frac{|a_j|}{A}$$

□

Pour profiter de l'information souple, il s'agit donc de modifier $P_d(h, y_\bullet)$ (et par suite $\Omega(S)$) en conséquence; si l'on reprend le calcul de $P_d(h, y_\bullet)$ en tenant compte du lemme ci-dessus, on trouve que (avec $y = (y_{10}|y_\bullet) = (y_1 ... y_N)$):

$$P_d(h, y_\bullet) =$$
$$\frac{2^{-(N-s)}\left(1 \Leftrightarrow 2^{-k} + 2^{-k}\prod_{i=1}^{N}\left(1 + (\Leftrightarrow 1)^{y_i}\prod_{j\in\mathrm{supp}(h)}\tanh\frac{|a_{ij}|}{A}\right)\right)}{(1\Leftrightarrow 2^{-k})(2^n \Leftrightarrow 1) + 2^{-k}\sum_{h'\in F_2^{n\star}}\prod_{i=1}^{p}\left(1 \Leftrightarrow \prod_{j\in\mathrm{supp}(h')}\tanh\frac{|a_{ij}|}{A}\right)\prod_{i=p+1}^{s}\left(1 + \prod_{j\in\mathrm{supp}(h')}\tanh\frac{|a_{ij}|}{A}\right)}$$

Et l'on ne saura malheureusement pas en calculer le dénominateur. Néanmoins, si l'on ne tient compte de l'information souple que pour les positions en dehors du code poinçonné, on trouve cette fois que:

$$P_d(h, y_\bullet) =$$
$$\frac{1 \Leftrightarrow 2^{-k} + 2^{-k}\left(1 \Leftrightarrow z^{\mathrm{wt}(h)}\right)^p\left(1 + z^{\mathrm{wt}(h)}\right)^{s-p}\prod_{i=s+1}^{N}\left(1 + (\Leftrightarrow 1)^{y_i}\prod_{j\in\mathrm{supp}(h)}\tanh\frac{|a_{ij}|}{A}\right)}{2^{N-s}\left((1 \Leftrightarrow 2^{-k})(2^n \Leftrightarrow 1) + 2^{-k}\sum_{w=1}^{n}\binom{n}{w}(1 \Leftrightarrow z^w)^p(1 + z^w)^{s-p}\right)}$$

Quantité que l'on saura calculer. On a alors:

$$\frac{P_d(h, y_\bullet)}{P_f(h, y_\bullet)} =$$
$$\frac{1 \Leftrightarrow 2^{-k} + 2^{-k}\left(1 \Leftrightarrow z^{\mathrm{wt}(h)}\right)^p\left(1 + z^{\mathrm{wt}(h)}\right)^{s-p}\prod_{i=s+1}^{N}\left(1 + (\Leftrightarrow 1)^{y_i}\prod_{j\in\mathrm{supp}(h)}\tanh\frac{|a_{ij}|}{A}\right)}{1 \Leftrightarrow 2^{-k} + \frac{2^{-k}}{2^n-1}\sum_{w=1}^{n}\binom{n}{w}(1 \Leftrightarrow z^w)^p(1 + z^w)^{s-p}}$$

L'ensemble $\Omega(S)$ et donc la fonction $\beta(S) = P_{det}(S, 1, 1)$ sont alors plus délicats à calculer. Cela posera un problème lorsque l'on voudra dimensionner correctement les paramètres S, m et x, besoin étant pour cela de connaître la fonction $\beta(S)$.

On sait cependant que celle-ci est décroissante et supérieure à celle obtenue sans tenir compte de l'information souple (car le critère est dans ce cas moins discriminant), ce qui pourra permettre d'effectuer un dimensionnement pessimiste; par ailleurs un dimensionnement par une méthode semi-empirique est toujours possible.

4.3.2 Cas général

Un générateur de moments de poids faible impose à hX^t d'avoir une certaine forme, et change la forme requise au fil des appels successifs. Il est également possible d'imposer une certaine forme à la matrice $\begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix} X^t$ et de générer ainsi directement des r⇔uplets de mots h. Il est surtout possible de définir un «post-filtrage» sur les moments générés en première main.

On génère ainsi m_1 mots $h_1 ... h_{m_1}$, de même que m_2 $(0 \leq m_2 \leq \binom{m_1}{2})$ couples de mots choisis parmi ces m_1 selon une certaine heuristique, ..., ainsi que m_r $(0 \leq m_r \leq \binom{m_1}{r})$ r⇔uplets de mots choisis parmi ces m_1 selon une certaine heuristique.

On adjugera alors pour la détection si au moins x_1 mots parmi les m_1 vérifient un certain critère du premier ordre, et/ou si x_2 couples parmi les m_2 vérifient un certain critère du second ordre, ..., et/ou si x_r r⇔uplets parmi les m_r vérifient un certain critère d'ordre r.

D'une manière plus générale, si on appelle $B(X)$ l'ensemble de tous les mots générés, de tous les couples sélectionnés,... et de tous les r⇔uplets sélectionnés, on calculera une certaine fonction F de cet ensemble, qui approxime le rapport $\frac{\Pr(B(\Lambda)=B(X)|\,\mathcal{H})}{\Pr(B(\Lambda)=B(X))}$; et l'on adjugera pour la détection si le résultat est supérieur à un certain seuil déterminant les probabilités de fausse alarme et de détection.

Les heuristiques et les critères que j'ai expérimentés ne permettent pas, suite aux sophistications successives qu'ils ont subies, de calculer explicitement, ni de borner de manière satisfaisante, ces probabilités. Ils ne sont pas optimaux, continuent et peuvent continuer encore longtemps à évoluer rapidement avec l'expérience acquise. C'est pourquoi je ne donne ni l'historique de ces évolutions, qui exigerait une interminable argumentation, ni l'état précis auquel j'en suis actuellement, dont la justification s'appuie sur cet historique.

L'objectif, de garantir (empiriquement si nécessaire) un couple (P_{fa}, P_{det}) donné pour des régions de plus en grandes de l'espace des tests $\{n, k, \tau, N\}$, à des coûts de plus en plus faibles, et par un algorithme sollicitant le moins possible l'expertise de l'opérateur, requiert une volumineuse étude et des prises de décision associées à un contexte précis.

Cette thèse consitue avant tout un point de départ et une base théorique pour une telle étude, ainsi qu'un recueil d'éléments qui devront y figurer, mais elle n'a pas correspondu à une demande réelle qui en aurait délimité les frontières; et le résultat final ne constitue pas vraiment un «produit fini».

4.3.3 Reconnaissance de longueur et de synchronisation

Nous supposerons maintenant que plusieurs hypothèses de longueur et de synchronisation différentes sont émises. Si l'on est sûr d'avoir le train binaire en entier, les longueurs de code possibles sont les diviseurs de la longueur du train binaire; et pour chaque longueur, on aura une unique hypothèse de synchronisation.

Si en revanche on ne dispose que d'une partie (connexe) du train, toutes les longueurs sont théoriquement possibles. Si l'on a une extrémité de ce train (qu'il s'agisse du premier ou du dernier bit), une seule hypothèse de synchronisation par longueur suffit. Dans le cas contraire, pour chaque longueur n, n hypothèses de synchronisation sont nécessaires.

Selon les cas, on aura donc un certain nombre l d'hypothèses $\mathcal{H}_1 \dots \mathcal{H}_l$ de longueur et de synchronisation, chacune pouvant être pondérée par une probabilité a priori que nous noterons $\Pr(\mathcal{H}_i \,|\, \mathcal{H})$. Si la longueur de l'hypothèse $\mathcal{H}_i$ est n, s'il y a x (en général $x = 1$ ou n) hypothèses de synchronisation associées à n, et si $\Pr(D_n|\,\mathcal{H})$ est la probabilité *a priori* pour que la longueur de l'hypothétique code soit n (cf. section 1.2.2), alors $\Pr(\mathcal{H}_i \,|\, \mathcal{H}) = \frac{\Pr(D_n \,|\, \mathcal{H})}{x}$.

L'hypothèse $\mathcal{H}$ consiste à dire que l'une de ces hypothèses est vraie. Si l'algorithme décide que $\mathcal{H}$ est vraie, il lui faudra ensuite chercher laquelle de ces hypothèses est la plus probable. Il aura alors effectué ce que l'on a appelé la reconnaissance de longueur et de synchronisation.

Appelons X le train binaire effectivement intercepté (il ne s'agit plus d'une matrice mais d'un vecteur) et Λ la variable aléatoire correspondante. Et pour chaque hypothèse $\mathcal{H}_i$, appelons n_i la longueur de code correspondante, N_i le nombre de mots reçus correspondant, X_i la $n_i \times N_i$-matrice extraite de X comme préconisé par l'hypothèse $\mathcal{H}_i$ et Λ_i la variable aléatoire correspondante. On a alors, pour tout X:

$$\Pr(\Lambda = X|\,\mathcal{H}) = \sum_{i=1}^{l} \Pr(\mathcal{H}_i \,|\, \mathcal{H}) \Pr(\Lambda = X|\,\mathcal{H}_i)$$

$$\frac{\Pr(\Lambda = X|\,\mathcal{H})}{\Pr(\Lambda = X)} = \sum_{i=1}^{l} \Pr(\mathcal{H}_i \,|\, \mathcal{H}) \frac{\Pr(\Lambda = X|\,\mathcal{H}_i)}{\Pr(\Lambda = X)}$$

Lorsque l'on extrait une matrice X_i de X, un certain nombre de bits sont laissés à part parce que n'entrant dans ancun mot de code entier. Nous supposerons que ces bits sont indépendants et de probabilité $1/2$ et l'on a dans ce cas pour tout i:

$$\frac{\Pr(\Lambda = X|\,\mathcal{H}_i)}{\Pr(\Lambda = X)} = \frac{\Pr(\Lambda_i = X_i|\,\mathcal{H}_i)}{\Pr(\Lambda_i = X_i)}$$

Pour chaque i, on choisira alors un certain algorithme B_i qui génèrera un ensemble $B_i(X_i)$ de mots, de couples de mots... Et on déterminera la fonction F_i adéquate qui à tout ensemble B associe une approximation du rapport $\frac{\Pr(B_i(\Lambda_i)=B|\,\mathcal{H}_i)}{\Pr(B_i(\Lambda_i)=B)}$. On prendra garde bien entendu à maîtriser les coûts de calcul.

On calculera alors $\sum_{i=1}^{l} \Pr(\mathcal{H}_i \,|\, \mathcal{H}) F_i(B_i(X_i))$.

L'algorithme décidera que $\mathcal{H}$ est vraie si cette quantité est supérieure à un certain seuil. Si tel est le cas, l'hypothèse de longueur et de synchronisation que l'on retiendra sera celle qui aura apportée la plus forte contribution (l'hypothèse $\mathcal{H}_i$ telle que $\Pr(\mathcal{H}_i \,|\, \mathcal{H}) F_i(B_i(X_i))$ est maximum).

On pourrait être tenté de tester les différentes hypothèses par ordre de longueur croissante, et de s'arrêter sitôt qu'une hypothèse s'avère crédible. Cependant, il est recommandé, pour la reconnaissance de longueur et de synchronisation, de tester toutes les hypothèses sans exception. En effet, il pourra arriver dans de nombreux cas qu'une mauvaise hypothèse soit *a priori* crédible, par exemple si elle ne diffère de la bonne hypothèse que par une légère erreur de synchronisation; il sera en revanche très rare, si le choix des B_i et F_i est fait correctement, qu'elle paraisse plus vraisemblable que la bonne hypothèse. Pour prévenir le risque de ne pas tester celle-ci, on les testera donc toutes.

Les cas les plus fréquents (de manière écrasante) de mauvaise reconnaissance seront alors les erreurs de synchronisation (la longueur est quant à elle plus facile à reconnaître). Une telle erreur peut être détectée simplement après la phase de reconnaissance de code à la forme de la matrice de parité obtenue. En effet, si la reconnaissance est de bonne qualité, cette matrice aura dans ce cas au moins une colonne de 0 à l'une de ses extrémités, c.a.d. que le support du code dual obtenu sera tronqué à cette extrémité; il sera alors facile de corriger l'erreur de synchronisation et de récupérer les calculs déjà effectués pour la reconnaissance (qu'il faudra terminer avec la nouvelle hypothèse de synchronisation de manière à compléter le support du code dual obtenu).

4.4 Reconnaissance de code

Désormais, nous considérons comme acquis le fait que l'hypothèse $\mathcal{H}$ est vraie, et que l'hypothèse de longueur et de synchronisation est la bonne, c.a.d. le fait que les lignes de la matrice correspondent à des mots de code erronés. Et nous supposons en outre que le code en question est de dimension comprise entre $k_{\min}$ et $k_{\max}$.

Il s'agit de reconstruire le code. Quel que soit l'algorithme choisi, il est raisonnable de supposer qu'il présente les caractéristiques suivantes:

- Il dispose d'un buffer de candidats h pour être des mots du code dual alimenté par le générateur de moments de poids faible. Puisque nous serons amenés, nous allons le voir, à calculer des combinaisons linéaires de ces mots, lesquelles peuvent constituer de nouveaux candidats, nous aurons éventuellement d'autres sources d'alimentation de ce buffer. Il sera de taille limitée et devra être composé de candidats aussi bons que possible.

- Un ou plusieurs candidats pour le code dual seront présents en mémoire sous forme de listes de mots h (le candidat pour le code dual est défini comme étant l'espace vectoriel engendré par ces mots).

- Tout candidat pour le code dual évoluera au fil des calculs. A l'instant initial, il vaudra l'espace vectoriel de dimension 0 (consitué uniquement du mot nul); a l'instant final, il vaudra un espace vectoriel de dimension comprise entre $n \Leftrightarrow k_{\max}$ et $n \Leftrightarrow k_{\min}$. A un instant donné, sa dimension sera donc un nombre r compris entre 0 et $n \Leftrightarrow k_{\min}$.

– L'algorithme disposera d'une routine chargée d'incrémenter d'une ou plusieurs unités la dimension du candidat.

Par ailleurs, il paraît souhaitable que l'algorithme soit capable d'une part de comparer la vraisemblance de deux candidats différents, d'autre part de décrémenter la dimension du candidat s'il semble qu'une mauvaise décision ait été prise.

La vraisemblance d'un candidat de dimension courante r ne peut pas être définie comme étant la probabilité pour que ce candidat soit égal au code dual, puisque l'on sait que ce n'est pas le cas si $r < n \Leftrightarrow k_{\max}$. Elle sera donc définie comme étant la probabilité pour qu'il soit inclus dans le code dual. Il s'agit donc de pouvoir calculer une fonction monotone en cette probabilité.

Si $h_1 ... h_r$ sont r mots indépendants du candidats, cette probabilité est directement liée à l'hypothèse $\mathcal{H}(h_1 ... h_r)$ selon laquelle les lignes de la matrice vérifiaient toutes, avant transmission, chacune des relations de parité définies par $h_1 \dots h_r$. Son calcul exact aura donc un coût plus qu'exponentiel en r. Pour être plus précis, ce coût vaut approximativement $N_H 2^r rn$, ainsi que nous l'avons vu en section 4.2.4, N_H étant défini comme étant le nombre de colonnes différentes dans la matrice HX^t.

On définit alors $r_{\max}$ comme étant la valeur maximale de r pour laquelle le calcul exact soit acceptable, c.a.d. pour laquelle un coût inférieur à $N_H 2^{r_{\max}} r_{\max} n$ est acceptable pour une sous-routine appelée un nombre polynomial en n de fois. Et nous supposerons que le coût d'un décodage relativement à n'importe quel code de longueur n, problème que, suite à la première partie de ce manuscrit, on sait bien bien optimiser, est inférieur à $2^{r_{\max}} r_{\max} n$ (pour des longueurs n trop grandes pour que le décodage ait un coût acceptable, les codes ne seront généralement pas détectables, donc encore moins reconnaissables).

Pour tout $r > r_{\max}$, la vraisemblance d'une hypothèse $\mathcal{H}(h_1 ... h_r)$ ne pourra qu'être estimée. Nous verrons que dès lors que r franchit ce seuil, il sera plus intéressant d'abandonner la représentation de sous-codes du code dual, pour passer à celle de surcodes du code lui-même.

Les représentants $h_1 ... h_r$ d'un candidat peuvent être *a priori* n'importe quels mots linéairement indépendants, éléments de ce candidat. Décrémenter la dimension d'un candidat ne doit donc pas consister simplement à supprimer l'un de ses représentants, mais à trouver, dans un cas idéal, le sous espace vectoriel de dimension $r \Leftrightarrow 1$ inclus dans le candidat, de vraisemblance maximale. Ce sous espace vectoriel peut très bien ne contenir aucun des mots $h_1 ... h_r$. Nous allons voir en section 4.4.3 comment calculer (une représentation de) ce sous-espace vectoriel.

4.4.1 Estimation d'une hypothèse sous contrainte de temps

Pour toute $r \times n$-matrice $H = \begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$, appelons P_H la probabilité $\Pr(\Lambda_N = X | \mathcal{H}(h_1 ... h_r))$ calculée en section 4.2.2:

$$P_H = 2^{-nN} \prod_{i=1}^{N} \sum_{b \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle b, x_i H^t \rangle} z^{\mathrm{wt}(bH)}$$

Et pour tout $b \in \mathbf{F}_2^r$, soient $N_{H,b}$ le nombre de colonnes de HX^t égales à b^t et $p_{H,b}$ la quantité $\sum_{b' \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b'H)}$ (on a ainsi $P_H = 2^{-nN} \prod_{b \in \mathbf{F}_2^r} {p_{H,b}}^{N_{H,b}}$).

Il suffit donc d'estimer les différents $p_{H,b}$ pour b^t parcourant l'ensemble des colonnes de HX^t, donc pour N_H valeurs distinctes de b. Le coût de l'estimation de $p_{H,b}$ doit être au plus $2^{r_{\max}} r_{\max} n$, et l'on suppose $r > r_{\max}$.

On ne pourra donc pas faire la sommation sur tous les b'. Mais l'on sait (corollaire 10) que si x est tel que $Hx^t = b^t$, et si $d = \min_{c \in \langle h_1 ... h_r \rangle^\perp} \mathrm{wt}(x+c)$ (d aura la même valeur pour tout x vérifiant $Hx^t = b^t$), alors $p_{H,b} > 2^r \tau^d (1 \Leftrightarrow \tau)^{n-d}$.

Cette minoration constituera l'estimation recherchée. On décodera donc, relativement au code $\langle h_1 \dots h_r \rangle^\perp$ un mot de syndrôme b (un mot x tel que $Hx^t = b^t$). Le poids du coset leader obtenu est le nombre d recherché. On définit alors les approximations $\tilde{p}_{H,b} = 2^r \tau^d (1 \Leftrightarrow \tau)^{n-d}$, et:

$$\tilde{P}_H = 2^{-nN} \prod_{b \in \mathbf{F}_2^r} \tilde{p}_{H,b}^{N_{H,b}}$$

En somme, lorsque r est trop grand, la méthode consiste en quelque sorte à abandonner la représentation duale pour travailler avec une représentation non duale du code candidat.

4.4.2 Incrémentation de la dimension courante

Soit $H = \begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$ une représentation du candidat $\mathrm{span}(h_1 ... h_r) = \mathrm{span}(H)$ courant.

Deux possibilités se présentent pour incrémenter la dimension du candidat.

La première consiste à trouver un mot h dans le buffer, n'appartenant pas à $\mathrm{span}(H)$, qui, dans un cas idéal, maximise la vraisemblance de $\mathrm{span}(h_1 ... h_r, h)$. Il est également possible de calculer la vraisemblance d'un nombre inférieur à la taille du buffer de candidats choisis aléatoirement. L'intérêt est que la procédure est plus rapide et non déterministe. Supposons pour simplifier que l'on doive parcourir le buffer entier.

Il nous faut un test d'appartenance à $\mathrm{span}(H)$. Pour cela, on calculera une matrice G duale de H. Et l'on a alors, pour tout h:

$$h \in \mathrm{span}(H) \Leftrightarrow Gh^t = 0$$

On marquera tous les h du buffer appartenant à span(H).

Soit alors, pour tout $h \notin \text{span}(H)$, $H_h = \begin{pmatrix} h_1 \\ \vdots \\ h_r \\ h \end{pmatrix}$. On calculera P_{H_h}.

Si l'on a calculé P_{H_h}, il est alors inutile de calculer $P_{H_{h'}}$ pour $h' \in \text{span}(H_h)$. On marquera donc également les h' du buffer appartenant à span(H_h). Remarquons que l'on a:

$$h' \in \text{span}(H_h) \Leftrightarrow (Gh'^t = 0 \text{ ou } Gh'^t = Gh^t)$$

On calculera successivement les vraisemblances P_{H_h} pour les h non marqués sachant que l'on a déjà calculé ou estimé P_H. Or:

$$\begin{aligned} P_{H_h} &= 2^{-nN} \prod_{i=1}^{N} \sum_{b \in \mathbf{F}_2^{r+1}} (\Leftrightarrow 1)^{\langle b, x_i H_h^t \rangle} z^{\text{wt}(bH_h)} \\ &= 2^{-nN} \prod_{i=1}^{N} \sum_{b \in \mathbf{F}_2^{r}} \left((\Leftrightarrow 1)^{\langle b, x_i H^t \rangle} z^{\text{wt}(bH)} + (\Leftrightarrow 1)^{\langle b, x_i H^t \rangle + \langle x_i, h \rangle} z^{\text{wt}(bH+h)} \right) \\ &= 2^{-nN} \prod_{i=1}^{N} \left(p_{H, x_i H^t} + (\Leftrightarrow 1)^{\langle x_i, h \rangle} \sum_{b \in \mathbf{F}_2^{r}} (\Leftrightarrow 1)^{\langle b, x_i H^t \rangle} z^{\text{wt}(bH+h)} \right) \end{aligned}$$

Si $r \leq r_{\max}$, P_H a été calculé, donc les termes ont été mémorisés. Il manque la moitié des termes, c.a.d. que pour compléter le calcul, il suffira de calculer les sommes $(\Leftrightarrow 1)^{\langle x_i, h \rangle} \sum_{b \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle b, x_i H^t \rangle} z^{\text{wt}(bH+h)}$.

Si $r > r_{\max}$, P_H n'a été qu'estimée. Le code $\langle h_1 \dots h_r, h \rangle^\perp$ est un sous-code de $\langle h_1 \dots h_r \rangle^\perp$. Si le mot du code $\langle h_1 \dots h_r \rangle^\perp$ obtenu par décodage de x_i appartient également à $\langle h_1 \dots h_r, h \rangle^\perp$, l'estimation de $p_{H, x_i H_h^t}$ sera la même que celle de $p_{H, x_i H^t}$; sinon il faudra calculer cette estimation.

Finalement, on choisira le h qui aura maximiser P_{H_h}.

La deuxième possibilité ne nécessite pas la consultation du buffer. Elle peut s'avérer plus intéressante que la première en fin de reconnaissance lorsque la dimension ne doit plus être incrémentée que d'une unité ou deux.

En effet, il arrive souvent lors de cette phase, que le buffer ne contienne pas de mots du dual n'appartenant pas déjà aux candidats, les particularités combinatoires des objets considérés et des régions explorées dans le temps imparti étant telles que seuls les moments correspondant aux éléments d'un sous-espace vectoriel strict du code dual aient été détectés.

Supposons donc que la dimension courante du candidat pour le code dual soit $r > r_{\max}$. A ce stade, on ne dispose pas seulement d'une base $(h_1 ... h_r)$ de ce candidat, mais également d'un ensemble de N mots $(c_1 ... c_N)$ obtenus par décodage des mots reçus $(x_1 ... x_N)$ relativement au code $\langle h_1 ... h_r \rangle^\perp$, supposé contenir strictement le code recherché.

Si l'on incrémente la dimension de $\langle h_1...h_r\rangle$ en y ajoutant un mot h, il faudra calculer un nouvel ensemble $(c'_1...c'_N)$ inclus dans $\langle h_1...h_r, h\rangle^\perp$. Pour tout i, si $\langle h, c_i\rangle = 0$, alors $c'_i = c_i$, sinon, il faut trouver $c'_i \in \langle h_1...h_r, h\rangle^\perp$ proche de x_i. Mais puisque $\langle h_1...h_r, h\rangle^\perp \subset \langle h_1...h_r\rangle^\perp$, on a $\mathrm{d}(c'_i, x_i) \geq \mathrm{d}(c_i, x_i)$.

Il paraît intéressant dès lors de décoder par liste: lorsque l'on décode x_i relativement à $\langle h_1...h_r\rangle^\perp$, on ne renvoie pas seulement le mot c_i optimal mais l'ensemble des mots de $\langle h_1...h_r\rangle^\perp$ trouvés en cours de décodage à distance inférieure à un seuil d_S de x_i. d_S devra être proche de $n\tau$, le nombre moyen d'erreur de transmission par mot.

Vers la fin de la reconnaissance, si le candidat est bien inclus dans le code dual, l'ensemble $(c_1...c_N)$ sera sûrement très proche de l'ensemble que l'on obtiendrait par décodage avec le bon code, les surcodes de celui-ci de dimension pas trop supérieure à la bonne dimension ne contenant probablement pas de mots plus proches des x_i que les mots de code transmis.

Il existe donc un ensemble très proche de $(c_1...c_N)$ de dimension au plus k. Réduire la dimension de $\langle c_1...c_N\rangle$ à un nombre au plus égal à $n \Leftrightarrow r \Leftrightarrow 1$ est équivalent à incrémenter celle de $\langle h_1...h_r\rangle$.

La dimension de $\langle c_1...c_N\rangle$ est au plus $n \Leftrightarrow r$. On cherchera un sous-ensemble J de $\{1...N\}$ de cardinal maximal, tel que la dimension de $\langle c_j; j \in J\rangle$ soit au plus $n \Leftrightarrow r \Leftrightarrow 1$. Et l'on décodera les mots x_j pour $j \notin J$ relativement au nouveau code $\langle c_j, j \in J\rangle$.

On pourra ainsi calculer la vraisemblance d'un nouveau candidat, si elle est satisfaisante, on calculera un mot h élément du dual du nouveau code, et n'appartenant pas à $\langle h_1...h_r\rangle$. Sinon, on essaiera un autre sous ensemble J de $\{1...N\}$ vérifiant la propriété requise.

Si l'on ne parvient pas ainsi à obtenir une incrémentation satisfaisante, c'est sans doute que la reconnaissance est terminée.

4.4.3 Décrémentation de la dimension courante

Soit $H = \begin{pmatrix} h_1 \\ \vdots \\ h_r \end{pmatrix}$ une représentation du candidat $\mathrm{span}(h_1...h_r) = \mathrm{span}(H)$. S'il s'agit de décrémenter la dimension de ce candidat, c'est que P_H a été calculée ou estimée et ne s'est pas avérée satisfaisante. Nous voudrions alors, dans un cas idéal, trouver le sous espace vectoriel de dimension $r \Leftrightarrow 1$ de ce candidat qui est le plus vraisemblablement inclus dans le code dual.

Or il existe une relation biunivoque entre l'ensemble des sous espaces vectoriels de dimension $r \Leftrightarrow 1$ de $\mathrm{span}(H)$ qui est de cardinal $2^r \Leftrightarrow 1$ (cf. [LN83], p. 455) et l'ensemble des vecteurs binaires non nuls de longueur r. On peut par exemple considérer la relation suivante qui est clairement biunivoque: pour tout $v \in \mathbf{F}_2^r$ non nul, l'ensemble $S_v \stackrel{def}{=} \{bH; b \in \mathbf{F}_2^r / \langle b, v\rangle = 0\}$ est un sous espace vectoriel de dimension $r \Leftrightarrow 1$ de $\mathrm{span}(H)$.

Pour tout $v \in \mathbf{F}_2^r$, on définit une $(r \Leftrightarrow 1) \times r$-matrice M_v génératrice de l'ensemble $\{bH; b \in \mathbf{F}_2^r / \langle b, v\rangle = 0\}$ de la manière suivante (v est une $1 \times r$

matrice de parité de cet ensemble): Soit i l'index de la dernière position non nulle de v, et soit $v' \in \mathbf{F}_2^{r-1}$ le vecteur obtenu à partir de v en supprimant la position i, la $i$ème colonne de M_v vaudra v'^t, les $r \Leftrightarrow 1$ autres colonnes de M_v constitueront une matrice diagonale. La $(r \Leftrightarrow 1) \times n$-matrice $M_v H$ constitue alors une représentation de S_v.

Il s'agit donc de trouver $\tilde{v} \in \mathbf{F}_2^r$ tel que la probabilité pour que $S_{\tilde{v}}$ soit inclus dans le code dual soit maximale, donc tel que $P_{M_{\tilde{v}}H}$ soit maximal.

Proposition 27 *Pour tout $v \in \mathbf{F}_2^r$ et tout $b \in \mathbf{F}_2^r$, on a:*

$$p_{M_v H, bM_v^t} = \frac{p_{H,b} + p_{H,b+v}}{2}$$

Démonstration: Par définition de $p_{M_v H, M_v b}$, on a:

$$p_{M_v H, M_v b} = \sum_{b' \in \mathbf{F}_2^{r-1}} (\Leftrightarrow 1)^{\langle b', M_v b \rangle} z^{\mathrm{wt}(b' M_v H)} = \sum_{b' \in \mathbf{F}_2^{r-1}} (\Leftrightarrow 1)^{\langle b' M_v, b \rangle} z^{\mathrm{wt}(b' M_v H)}$$

Or $\{b' M_v ; b' \in \mathbf{F}_2^{r-1}\} = \{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 0\}$, donc:

$$p_{M_v H, M_v b} = \sum_{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 0} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)}$$

Par ailleurs, on a:

$$\begin{aligned} p_{H,b} &= \sum_{b' \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)} \\ &= \sum_{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 0} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)} + \sum_{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 1} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)} \end{aligned}$$

$$\begin{aligned} p_{H,b+v} &= \sum_{b' \in \mathbf{F}_2^r} (\Leftrightarrow 1)^{\langle b', b+v \rangle} z^{\mathrm{wt}(b' H)} \\ &= \sum_{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 0} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)} \Leftrightarrow \sum_{b' \in \mathbf{F}_2^r / \langle b', v \rangle = 1} (\Leftrightarrow 1)^{\langle b', b \rangle} z^{\mathrm{wt}(b' H)} \end{aligned}$$

On en déduit la proposition. □

Proposition 28 *Pour tout $v \in \mathbf{F}_2^r$, on a:*

$$P_{M_v H} = 2^{-nN} \prod_{b \in \mathbf{F}_2^r} \left(\frac{p_{H,b} + p_{H,b+v}}{2} \right)^{N_{H,b}}$$

Démonstration: On a:

$$P_{M_vH} = 2^{-nN} \prod_{b \in \mathbf{F}_2^{r-1}} {p_{M_vH,b}}^{N_{M_vH,b}}$$

On sait que $vM_v^t = 0$, donc pour tout $b \in \mathbf{F}_2^r$ on a $bM_v^t = (b+v)M_v^t$. Si on énumère les $bM_v^t \in \mathbf{F}_2^{r-1}$ pour $b \in \mathbf{F}_2^r$, on comptera deux fois chaque élément de $\mathbf{F}_2^{r-1}$.

On en déduit que $\prod_{b \in \mathbf{F}_2^r} {p_{M_vH,bM_v^t}}^{N_{M_vH,bM_v^t}} = (2^{nN} P_{M_vH})^2$.

Or $N_{M_vH,bM_v^t} = N_{H,b} + N_{H,b+v}$ car, quel que soit $x \in \mathbf{F}_2^n$, on a:

$$(M_vH)x^t = M_vb^t \Leftrightarrow (Hx^t = b^t \text{ ou } Hx^t = (b+v)^t)$$

Donc:

$$\begin{aligned}
(2^{nN} P_{M_vH})^2 &= \prod_{b \in \mathbf{F}_2^r} {p_{M_vH,bM_v^t}}^{N_{H,b}+N_{H,b+v}} \\
&= \prod_{b \in \mathbf{F}_2^r} \left(\frac{p_{H,b} + p_{H,b+v}}{2} \right)^{N_{H,b}+N_{H,b+v}} \\
&= \prod_{b \in \mathbf{F}_2^r} \left(\frac{p_{H,b} + p_{H,b+v}}{2} \right)^{N_{H,b}} \prod_{b \in \mathbf{F}_2^r} \left(\frac{p_{H,b} + p_{H,b+v}}{2} \right)^{N_{H,b+v}} \\
&= \left(\prod_{b \in \mathbf{F}_2^r} \left(\frac{p_{H,b} + p_{H,b+v}}{2} \right)^{N_{H,b}} \right)^2
\end{aligned}$$

Les quantités en jeu étant toutes positives, on en déduit l'égalité proposée. □

Nous sommes maintenant mieux préparé à trouver $\tilde{v} \in \mathbf{F}_2^r$ tel que $P_{M_{\tilde{v}}H}$ soit maximale.

Supposons d'abord que $r \leq r_{\max}$, tout calcul de coût inférieur à $N_H 2^r rn$ est alors envisageable. Si un calcul de coût $2^{2r}rn$ l'est aussi, on mettra alors en mémoire tous les $p_{H,b}$ ($b \in \mathbf{F}_2^r$) (remarquons que puisque $r \leq r_{\max}$, P_H a été calculé, ainsi qu'une partie des $p_{H,b}$, si ce n'est tous, il semble opportun de les mémoriser à ce moment).

Pour trouver $\tilde{v} \in \mathbf{F}_2^r$ qui maximise le produit $\prod_{b \in \mathbf{F}_2^r} (p_{H,b} + p_{H,b+v})^{N_{H,b}}$, on pourra donc tester chaque $v \in \mathbf{F}_2^r$, le coût de calcul dudit produit étant alors majoré par N_H additions et multiplications.

Si le coût maximal est compris entre $N_H 2^r rn$ et $2^{2r}rn$, on se reportera au cas $r > r_{\max}$ ou bien on procèdera comme suit.

On dispose en tout cas de N_H des $p_{H,b}$, suite au calcul de P_H. On estimera les autres et l'on mettra tout cela en mémoire. Il sera alors encore envisageable de tester chaque $v \in \mathbf{F}_2^r$, le coût de l'estimation de $\prod_{b \in \mathbf{F}_2^r} (p_{H,b} + p_{H,b+v})^{N_{H,b}}$

étant majoré par N_H additions et multiplications (on ne s'occupe pas des b tels que $N_{H,b} = 0$).

Supposons maintenant $r > r_{\max}$, tout calcul de coût supérieur à $M2^r rn$ est alors inenvisageable. En particulier, P_H n'a pu être qu'estimé.

Tant que le nombre total de $p_{H,b}$ estimés est très inférieur à 2^r, les collisions sont peu probables et chaque v que l'on voudra tester demandera l'estimation d'environ N nouveaux $p_{H,b}$. La proposition 28 ne peut donc pas rendre le test d'un v beaucoup moins coûteux que l'estimation de P_H; le calcul direct, d'ordre $r \Leftrightarrow 1$, de $P_{M_v H}$ paraît donc préférable.

Il s'agira de toute façon de limiter le nombre des tests. Appelons $\tilde{v}$ le v optimal que nous aimerions trouver. De part la forme de la matrice $M_{\bar{v}}$, les mots h_i $(1 \leq i \leq r)$ figurent en ligne dans la matrice $M_{\bar{v}}H$ si et seulement $\tilde{v}_i = 0$. On en déduit, puisque $\tilde{v}$ est optimal, que les sous-espaces vectoriels que l'on peut former avec les combinaisons linéaires des mots h_i tels que $\tilde{v}_i = 0$ présentent une vraisemblance en principe plus élevée que celles que l'on obtiendrait en y mélant des h_i tels que $\tilde{v}_i = 1$.

Le calcul des vraisemblances de candidats $\mathrm{span}(h_j, j \in J)$ $(J \subset \{1...r\})$ peut donc apporter de l'information sur la valeur de $\tilde{v}$.

On appliquera une procédure d'incrémentation (non déterministe) pour trouver un sous-ensemble J maximal de $\{1...r\}$ ayant une forte vraisemblance; et l'on testera le v de support le complémentaire de J. On répètera l'opération un certain nombre de fois et l'on gardera le meilleur v.

Si $\tilde{v}$ est finalement le $r \Leftrightarrow$uplet retenu, on calculera, si ce n'est déjà fait, la représentation $M_{\bar{v}}H$ et la vraisemblance du nouveau candidat. Si cette vraisemblance est satisfaisante on passera la main à l'incrémentation, sinon on décrémentera à nouveau.

4.4.4 Exemple d'algorithme

L'algorithme dispose d'un buffer de taille T aussi élevée que le permet la mémoire disponible, contenant des couples (h, hX^t), et initialement vide.

Nous supposerons qu'un générateur de moments de poids faibles tourne en permanence sur un processeur parallèle, et alimente le buffer chaque fois qu'il trouve un couple (h, hX^t) de vraisemblance $(1 \Leftrightarrow z^{\mathrm{wt}(h)})^{\mathrm{wt}(hX^t)}(1+z^{\mathrm{wt}(h)})^{N-\mathrm{wt}(hX^t)}$ supérieure à un seuil S assez bas pouvant être mis à jour une fois le buffer plein.

Les éléments du buffer sont triés de deux manières différentes: par ordre lexicographique sur h et par vraisemblance décroissante. Lorsque le buffer doit être libéré de l'un de ses éléments pour laisser la place à un nouveau, on choisit celui qui minimise la vraisemblance.

On considère $n \Leftrightarrow k_{\min}$ familles F_i $(i = 1...n \Leftrightarrow k_{\min})$ de candidats. Pour chaque i, la famille F_i contient un nombre m_i de candidats de dimension i. On calcule la vraisemblance de chaque candidat. Les candidats sont initialement vides. Les couples (h, hX^t), combinaisons linéaires d'éléments du buffer, générés par les calculs de vraisemblance viennent eux aussi alimenter le buffer. Par ailleurs, par souci d'économie, les calculs de vraisemblance puisent dans le buffer, lorsqu'ils

y figurent, les couples (h, hX^t) dont ils ont besoin (ceci justifie le tri par ordre lexicographique et la grande taille du buffer).

Pour $i \leq r_{\max}$, un candidat $H \in F_i$ est représenté par la totalité des couples (h, hX^t) avec $h \in H$, les i premiers couples $(h_1, h_1X^t)...(h_i, h_iX^t)$ étant tels que $h_1...h_i$ constitue une base sous forme échelon du candidat.

Pour $i \geq r_{\max}$, un candidat $H \in F_i$ est représenté par une base $(h_1...h_i)$ sous forme échelon, et pour tout $j = 1...N$, par un ensemble C_i^j de mots de $\langle h_1...h_i \rangle^\perp$ à distance inférieure à $K_i n\tau$ de x_j, K_i sera proche de 1 et sera à paramétrer (pour $i = r_{\max}$, les deux représentations sont utilisées).

Le buffer peut être considéré comme étant la famille F_1. Toute famille F_i est triée par vraisemblance (calculée ou estimée) décroissante.

Pour construire un nouveau candidat dans F_j $(j \geq 2)$, on choisit simplement un élément de F_{j-1} de manière aléatoire pondérée par la vraisemblance et, de la même manière, un élément de F_1 n'appartenant pas au précédent. Les familles sont continuellement mises à jour, leurs éléments les moins vraisemblables étant libérés au fur et à mesure.

Au bout d'un effort de calcul paramétrable, raisonnable mais en rapport avec la difficulté du problème (dépendant de n et de τ), on arrête les mises à jour. On essaye alors d'incrémenter par la deuxième méthode la dimension des candidats de dimension supérieure ou égale à $n \Leftrightarrow k_{\max} \Leftrightarrow 3$. Finalement, on retourne le candidat de dimension comprise entre $n \Leftrightarrow k_{\max}$ et $n \Leftrightarrow k_{\min}$ qui maximise la vraisemblance.

Conclusion

Ce long chapitre termine notre étude. Il y est défini une solution aux problèmes posés, valable pour une part importante des «cas réels». Ou plutôt, nous avons vu en quoi devait consister une telle solution, nous avons fait l'inventaire des outils à utiliser, des dangers à éviter et nous avons illustré ces considérations par des exemples d'algorithmes... Mais les «cas réels» sont spécifiques à un contexte, les solutions dépendront également de la puissance de calcul disponible, et nous ne saurions proposer une solution générale.

Pour finaliser un produit, les exemples d'algorithmes que nous avons donnés pourraient s'avérer suffisants, à condition toutefois de dimensionner convenablement tous les paramètres laissés en suspens. Pour ce faire, la théorie se révèle inadéquate, l'observation, le bon sens et l'intuition permettant d'améliorer les algorithmes à une vitesse trop élevée pour que la théorie suive.

L'implémentation des algorithmes et le dimensionnement des paramètres nécessiteront une étude et une batterie de tests volumineuses, qui ne relèvent guère de la science. Cependant, tout est donné ici pour les mener à bien.

Les implémentations testées, correspondent au cas défavorable où l'erreur de transmission est répartie de manière homogène sur le train binaire, où le code utilisé est aléatoire ainsi que les messages à encoder et où l'on ne connaît ni la longueur ni la synchronisation. Il s'est avéré que la probabilité de détecter une structure dans le train binaire en moins de quelques dizaines de secondes

de calcul, et avec une probabilité de fausse alarme (empirique) négligeable, est assez élevée ($> 90\%$) dans les conditions suivantes:

Le code doit être de longueur inférieur à 1000. Les mots de code doivent contenir en moyenne moins de $K(R)$ erreurs, $K(R)$ dépendant du taux de codage et ne dépendant pas ou très peu de la longueur. On a relevé les valeurs suivantes: $K(0.2) \approx 6$, $K(0.5) \approx 3$, $K(0.8) \approx 1.5$.

La probabilité de reconnaître le code semble suivre la même règle avec des valeurs différentes pour $K(R)$: $K(0.2) \approx 3.5$, $K(0.5) \approx 2.2$, $K(0.8) \approx 1.2$.

L'erreur la plus fréquente porte sur la dimension du code reconnu: Il manque une relation de parité.

Conclusion

Nous avons mis en perspective les différents problèmes posés par la détection et la reconnaissance de code. On en viendra à bout de manière d'autant plus sûre que l'on a une bonne connaissance des probabilités *a priori* auxquelles obéissent les sources possibles du train binaire.

Nous avons montré que la réduction de rang, problème plus simple (car on suppose que la longueur, la synchronisation et la dimension ne peuvent prendre qu'une valeur) que la détection optimale ou la reconnaissance à maximum de vraisemblance de code était NP-complet.

De fait, on ne saura résoudre ces problèmes qu'à la condition que le nombre moyen d'erreurs affectant un mot de code soit majoré par une constante (ne dépendant pas de n).

On a très peu parlé du fait qu'il est souhaitable de travailler sur une portion «propre» du train binaire, et de la détection ou de la reconnaissance dans le cas où le train binaire est dépourvu d'erreur. Dans ce cas de toute façon, le problème posé est trivial.

Il nécessite cependant de considérer une portion connexe et dépourvue d'erreurs du train binaire, de longueur évoluant comme le carré de la longueur de bloc du code utilisé. Or les codes linéaires binaires utilisés présentent généralement une longueur de bloc supérieure à 100 (ce qui rend le problème fort différent de la détection de code convolutif). Et il est alors fort peu probable qu'une portion connexe du train binaire de longueur convenable soit totalement dépourvue d'erreur.

Si l'on considère que le train binaire est entâché d'un petit nombre d'erreurs, la longueur de train binaire nécessaire pour résoudre le problème est alors augmentée, ainsi par conséquent que le nombre d'erreurs, et ainsi de suite... Finalement, il faudra bien se ramener au cas où l'on considère un nombre moyen d'erreurs par mot de code.

Une étude relevant de la théorie des codes reste à faire: la reconnaissance de structure dans un code reconnu: s'agit-il d'un code de Goppa, d'un BCH raccourci ou poinçonné...? Peut-on s'en rendre compte par simple examen d'une matrice génératrice?

Les codes q-aires, avec q une puissance de 2, sont souvent utilisés sous forme de codes étalés (un symbôle $q\Leftrightarrow$aire, avec $q = 2^m$, est transmis sous forme de m bits, le code étalé est un code linéaire binaire). Pour des erreurs réparties de manière homogène sur les bits, ils sont généralement moins intéressants que les

codes binaires, mais ils permettent de corriger les «bursts» d'erreur, lesquels se concentrent sur un nombre restreint de symbôles $q \Leftrightarrow$aires reconstitués à partir des bits, et sont en outre rapidement décodables. Pour ces raisons les codes utilisés sont souvent des codes étalés.

Le code reconnu est-il un code étalé? Quelle est la taille de l'alphabet, la synchronisation des symbôles? Le code $q \Leftrightarrow$aire correspondant est-il un code géométrique, un Reed-Solomon généralisé...

Il est possible d'apporter une réponse (par un algorithme) à la plupart de ces questions (je ne l'ai pas fait moi-même part manque de temps). Elles relèvent également du problème de détection et de reconnaissance de code. Elles demandent une certaine expertise en théorie des codes, et un étudiant de troisième cycle spécialisé dans ce domaine pourra en quelques mois de travail résoudre un certain nombre d'entre elles...

Enfin, un problème d'intérêt majeur, pour que la détection et la reconnaissance puissent capturer une part beaucoup plus importante des cas réels, serait de pouvoir traiter le cas des codes entrelacés.

Il m'a semblé que le cas général était hors de portée de l'informatique: l'entrelacement porte sur de grandes longueurs de train binaire, il doit être inversé avant la détection, or je n'ai trouvé aucune heuristique reconstruisant en temps raisonnable le bon entrelacement et le nombre de possibilités excèdent les complexités algorithmiques acceptables... Et même si l'on pouvait reconstituer des mots de code, d'une part ceux-ci seraient dans un ordre aléatoire, d'autre part, et c'est plus ennuyeux, on ne pourrait reconnaître le code que modulo ses permutations.

Une solution peu satisfaisante pourrait consister à faire l'inventaire des entrelacements normalisés et à tester ces derniers, ou bien à considérer une famille d'entrelacements particuliers... Peut-être y-aurait-il quelque fructueuse méthode si l'on abandonnait l'hypothèse que les bits de la source valent 0 ou 1 avec la même probabilité et sont indépendants...

Conclusion générale

Il a été question de décodage dans un premier temps, puis de détection et de reconnaissance, le tout appliqué exclusivement aux codes linéaires binaires, et sans accorder de traitement particulier à quelqu'un d'entre eux.

Shannon a déterminé l'efficacité maximale d'un codage de canal (fonction du canal, du taux de codage et de la longueur de code); il s'avère qu'elle coïncide avec l'efficacité moyenne pour une plage importante des paramètres. Il paraît donc peu probable que l'on puisse définir une famille de codes linéaires binaires à taux constant, combinatoirement exceptionnels, qui offrent un pouvoir de correction supérieur à la moyenne.

En dehors du troisième chapitre, l'étude entreprise sur le décodage est une synthèse qui devrait amener le lecteur à se convaincre que la correction d'erreur se repose sur les propriétés «banales» des codes linéaires binaires, et non sur des exceptions combinatoires, et que si les exceptions combinatoires étudiées par les mathématiciens permettent de définir des algorithmes de décodage efficaces, elles offrent en revanche un pouvoir de correction d'erreur généralement inférieur à celui qu'ont les codes aux propriétés combinatoires moyennes, et qui sont difficiles à décoder.

Cette étude permettra aussi au lecteur de s'approprier les techniques algorithmiques permettant de réduire la complexité du décodage de codes linéaires binaires aléatoires, et de mesurer la complexité minimale de la résolution de ce problème par les techniques connues à ce jour. On constate encore la supériorité de l'aléatoire, auquel les algorithmes les plus efficaces ont largement recours.

Le troisième chapitre, traitant du décodage souple, apporte quant à lui des éléments introuvables dans la littérature, et montrent le double intérêt de l'information souple: augmenter le pouvoir de correction et réduire la complexité du décodage. Les perspectives ouvertes par les solutions proposées au décodage souple quasi-optimal sont importantes car les transmissions numériques les plus courantes pourraient en bénéficier.

Nous avons ensuite étudié les problèmes de détection et de reconnaissance. L'étude probabilistique sous-jacente, malgré les efforts déployés pour la clarifier et la ponctuer de considérations à la lecture moins fatigante, est volumineuse et trouble. Il est apparu plus intéressant de la corroborer par des simulations que de la rendre parfaitement consistante (et du même coup beaucoup plus volumineuse).

Bien que très différents du décodage, nous avons vu que la résolution opti-

male de ces problèmes exigeait des techniques algorithmiques très proches de celles utilisées pour le décodage (diagonalisations, poinçonnage, séparation des syndrômes, tris, boîtes...). Par ailleurs, la reconnaissance de code utilise abondamment le décodage par les codes candidats.

Ces deux problèmes sont d'autant plus difficiles que le code utilisé est long. Les solutions proposées fonctionnent parfaitement, en temps raisonnable, et avec des probabilités de détection et de fausse alarme acceptables, tant que le nombre moyen d'erreurs par mot de code reste majoré par un certain nombre (décroissant avec le taux de codage, depuis quelques unités jusqu'à zéro), indépendant de la longueur.

Une fois encore, le non-déterminisme et le pseudo-aléa sont primordiaux et irréductibles pour une bonne résolution de ces problèmes. Cette thèse illustre donc de manière assez éloquente l'importance de l'aléa, qui est actuellement au coeur des sciences de l'informatique et de la complexité, et se révèle indispensable ou du moins utile à l'accomplissement automatisé de bien des taches où on ne l'attendait pas.

Bibliographie

[Bar98] A. Barg. *Handbook of Coding Theory*, chapter Complexity issues in coding theory, pages 649–754. Elsevier, 1998.

[BC91] N. Balakrishnan and A. Clifford Cohen. *Ordered statistics and inference: Estimation methods.* CA: Academic Press, San Diego, 1991.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error correcting coding and decoding: Turbo-codes. In *Proc. Int. Conf. Communications*, pages 1064–1070, Geneva, Switzerland, May 23-26 1993.

[BKvT99] A. Barg, E. Krouk, and H. C. A. van Tilborg. On the complexity of minimum distance decoding of long linear codes. *IEEE Transactions on Information Theory*, 45(5):1392–1405, July 1999.

[Bli87] V.M. Blinovskii. Lower asymptotic bound on the number of linear code words in a sphere of given radius in f_q^n. *Probl. Inform. Transm.*, 23(2):50–53, 1987. in Russian, English Translation pp 130-132.

[BMP87] A. Benveniste, M. Metivier, and P. Priouret. *Algorithmes adaptatifs et approximations stochastiques.* Masson, 1987.

[CB97] F. Chapeau-Blondeau. Noise-enhanced capacity via stochastic resonance in an asymetric binary channel. *Physical Review*, E55:2016–2019, 1997.

[CC81] G. Clark and J. Cain. *Error correcting coding for digital communication.* Plenum, London, U.K., 1981.

[CC94] A. Canteaut and H. Chabanne. A further improvement of the work factor in an attempt at breaking McEliece cryptosystem. In P. Charpin, editor, *Livre des résumé – EUROCODE 94*, pages 163–167. INRIA, October 1994.

[CC95] A. Canteaut and F. Chabaud. Improvements of the attacks on cryptosystems based on error-correcting codes. Technical Report LIENS-95-21, Ecole Normale Supérieure, July 1995.

[CC98] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, January 1998.

[CG81] A. H. Chan and R. A. Games. (n, k, t)-covering system and error-trapping decoding. *IEEE Transactions on Information Theory*, IT-27:643–646, 1981.

[CG90] J. T. Coffey and R. M. Goodman. The complexity of information set decoding. *IEEE Transactions on Information Theory*, 31:1031–1037, Sep 1990.

[DK00] I.I. Dumer and R. Krichevskiy. Soft-decision majority decoding of reed-muller codes. *IEEE Transactions on Information Theory*, 46(1):256–264, Jan 2000.

[Dor74] B. G. Dorsch. A decoding algorithm for binary block codes and j-ary output channels (corresp.). *IEEE Transactions on Information Theory*, 20(3):391–394, May 1974.

[Dum89] I.I. Dumer. Two decoding algorithms for linear codes. *Probl. Inform. Transm.*, 25(1):17–23, 1989. English translation.

[Dum91] I.I. Dumer. On mimimum distance decoding of linear codes. In *5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, 1991. Moscow, USSR.

[Dum93] I.I. Dumer. Suboptimal decoding of linear codes. In *EUROCODE 92*, number 339 in CISM Courses and Lectures, pages 369–382. Springer-Verlag, 1993.

[Dum96a] I.I. Dumer. Covering lists in maximum-likelihood decoding. In *34th Annu. Allerton Conf. Communication, Control and Computing*, pages 683–692, 1996.

[Dum96b] I.I. Dumer. Suboptimal decoding of linear codes: Partition technique. *IEEE Transactions on Information Theory*, 42:1971–1986, 1996.

[Dum99] I.I. Dumer. Sort-and-match algorithm for soft decision decoding. *IEEE Transactions on Information Theory*, 45(7):2333–2338, nov 1999.

[Dum01] I.I. Dumer. Soft decision decoding using punctured codes. *IEEE Transactions on Information Theory*, 2001. to be published, 22 pp.

[Eli55] P. Elias. Coding for noisy channel. *IRE Conv. Rec.*, pages 37–46, 1955.

[ES74] P. Erdos and J. Spencer. *Probabilistic methods in combinatorics*. Akademiai Kiado, Budapest, 1974.

[Evs83] G.S. Evseev. Complexity of decoding for linear codes. *Probl. Inform. Transm.*, 19(1):1–6, 1983. english translation.

[FCGB88] J. Fang, G. Cohen, P. Godlewski, and G. Battail. On the inherent intractability of soft decision decoding of linear codes. In G. Cohen and P. Godlewski, editors, *Proc. of the Second Int. Col. on Coding Theory held in Cachan, November 24-26, 1986*, volume 311 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988.

[FL95] M. P. C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, Sept 1995.

[FL96a] M. P. C. Fossorier and Shu Lin. Computationally efficient soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 42(3):738–750, Mai 1996.

[FL96b] M. P. C. Fossorier and Shu Lin. First-order approximation of the ordered binary-symmetric channel. *IEEE Transactions on Information Theory*, 42(5):1381–1387, Sept 1996.

[FL97] M. P. C. Fossorier and Shu Lin. Complementary reliability-based decodings of binary linear block codes. *IEEE Transactions on Information Theory*, 43(5):1667–1672, Sept 1997.

[FL99] M. P. C. Fossorier and Shu Lin. Reliability-based information set decoding of binary linear codes. *IEICE Transactions on Fundamentals*, E82-A:2034–2042, Oct 1999.

[Gal62] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8:21–28, Jan 1962.

[Gal68] R. G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, 1968.

[GCB97] X. Godivier and F. Chapeau-Blondeau. Stochastic resonance in the information capacity of a nonlinear dynamic system. *Int. Journal of Bifurcation and Chaos*, 8(3):581–589, 1997.

[Gol49] M. J. E. Golay. Notes on digital coding. *Proc. IRE (Corresp.)*, 37:657, Juin 1949.

[GS97] D. Gazelle and J. Snyders. Reliability-based code-search algrithms for maximum-likelihood decoding of block codes. *IEEE Transactions on Information Theory*, 43(1):239–249, Jan 1997.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29:147–160, 1950.

[Han98] Y. S. Han. A new decoding algorithm for complete decoding of linear block codes. *SIAM J. Discrete math.*, 11(4):664–671, Nov 1998.

[HHC94] Y. S. Han, C. R. P. Hartmann, and C. C. Chen. Efficient priority first search maximum-likelihood soft decision decoding of linear block codes. *IEEE Transactions on Information Theory*, 39(5):1514–1523, Sept 1994.

[Hwa79] T.Y. Hwang. Decoding linear block codes for minimizing word error rate. *IEEE Transactions on Information Theory*, 25(6):733–737, 1979.

[Jr66] G. D. Forney Jr. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, IT-12(2):125–131, Apr 1966.

[Kis96] L. B. Kiss. *Chaotic, Fractal and Nonlinear Signal Processing*, chapter Possible breakthrough: Significant improvement of signal to noise ratio by stochastic resonance, pages 382–387. AIP Press, 1996.

[Kro89] E. A. Krouk. Decoding complexity bound for linear block codes. *Probl. Inform. Transm.*, 25(3):251–254, 1989. english translation.

[LB88] P.J. Lee and E.F. Brickell. An observation on the security of mceliece's public key cryptosystem. In G. Goos and J. Hartmanis, editors, *Advances in cryptology-EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer-Verlag, 1988.

[LGK96] K. Loerincz, Z. Gingl, and L. B. Kiss. A stochastic resonator is able to greatly improve signal to noise ratio. *Physical Letter*, A224:63–67, 1996.

[LH85] L. Levitin and C. Hartmann. A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm. *IEEE Transactions on Information Theory*, IT-31:378–384, 1985.

[LN83] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, 1983.

[LV95] A. Lafourcade and A. Vardy. Lower bounds on treillis complexity of block codes. *IEEE Transactions on Information Theory*, 41:1938–1952, 1995.

[Man80] D. Mandelbaum. On complete decoding of linear error-correcting codes. *Inform. and Control*, 47:195–200, 1980.

[McE78] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN progress report 42-44, Jet propulsion Lab. Calif. Inst. of Tech., 1978. pp 114-116.

[MPO94] F. Moss, D. Pierson, and D. O'Gormanm. Stochastic resonance: Tutorial and update. *Int. Journal of Bifurcation and Chaos*, 4:1383–1398, 1994.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[Pla96] G. Planquette. *Identification de trains binaires codés*. Thèse de doctorat, Université de Rennes 1, December 1996.

[Pol94] G. Poltyrev. Bounds on the decoding error probability of binary linear codes via their spectra. *IEEE Transactions on Information Theory*, 40(4):1284–1292, Jul 1994.

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inform. Theory*, IT-8:S5–S9, 1962.

[SGB67] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp. Lower bounds to error probability for coding on discrete memoryless channels. *Inform. and Control*, 10:65–103 (Part I), 522–552 (Part II), 1967.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423 and 623–656, 1948.

[Sha59] C. E. Shannon. Probability of error for optimal codes in a gaussian channel. *Bell System Tech. J.*, 38:611–656, 1959.

[Sle56] D. Slepian. A class of binary signaling alphabets. *Bell System Tech. J.*, 35:203–234, 1956.

[Sol93] P. Solé. *Moments d'Ordre Supérieur en Combinatoire Algébrique : Codes, Graphes, et Configurations*. PhD thesis, Université de Nice - Sophia Antipolis, nov 1993.

[Ste89] J. Stern. A method for finding codewords of small weight. In *Coding theory and applications (Toulon, 1988)*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1989.

[Sud93] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems*. PhD thesis, University of California, 1993.

[TP91] Dana J. Taipale and Michael B. Pursley. An improvement to generalized-minimum-distance decoding (corresp.). *IEEE Transactions on Information Theory*, 37(1):167–172, Jan 1991.

[Var97] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43:1757–1766, November 1997.

[vT94] J. van Tilburg. *Security-analysis of a class of cryptosystems based on linear error-correcting codes.* Netherland, Univ. Eindhoven, 1994. ISBN 90-72125-45-2.

[WM95] K. Wiesenfield and F. Moss. Stochastic resonance and the benefits of noise: From ice ages to crayfish and squids. *Nature*, 373:33–36, 1995.

[YJ80] Christopher Chi-Hsun Yu and Daniel J. Costello Jr. Generalized minimum distance decoding algorithms for $q$ary output channels (corresp.). *IEEE Transactions on Information Theory*, IT-26(2):238–243, Mar 1980.

[ZS93] V. V. Zyablov and V. R. Sidorenko. Bounds on complexity of treillis decoding of linear codes. *Probl. Transm. Inform.*, 29(3):3–9, 1993.

Abstract

Three problems, related to the transmission of a numeric signal and to error correcting codes, are treated. We only study the case of a binary stream (possibly with soft information), and of binary linear codes.

The first and most usual one, the decoding problem, has been abundantly treated in the past, and from very different points of view. Our main goal is the best possible utilization of the channel. The decoding complexity comes immediately after.

For this reason, we study long general (random) binary linear codes, which presents the best spectral properties, and their near-maximum-likelihood decoding, nearing the lowest error rates while faster than complete decoding.

The ambition of designing *the fastest algorithm* succeeding in such a decoding has continually guided the bibliographical choices and the approach. Eventually, an algorithm is proposed which actually seems to be the the fastest one so far.

On the other hand, the two other problems: the detection and the recognition of a code, are original, very little literature having been hallowed to them. They consist, observing a binary stream outcoming from a noisy channel, in deciding wether a binary linear code has been used to carry the signal (detection), and in case of positive decision, in determining the code in question (recognition).

We prove the NP-completeness of the subsequent decision problem. We then consider a broad class of problems comprising those we are interested in: the detection and recognition problems. And we study what should be the right way to solve them.

Lastly, we design solutions with a concern for optimality and generality, as well as a certain pragmatism. But these solutions are perfectible by nature, and are suggested only once the tools are given, that allow one to design his own solution.