N° d'ordre: 2821
de la thèse

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE RENNES 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention :Informatique

PAR

Thomas Guionnet

Équipe d'accueil : TEMICS/IRISA/INRIA
École Doctorale : Mathématiques, Télécommunications, Informatique, Systèmes Electroniques (MATISSE)
Composante universitaire : Institut de Formation Supérieure en Informatique et Communication

TITRE DE LA THÈSE :

Codage robuste par descriptions multiples pour transmission sans fil d'information multimédia

SOUTENUE LE 19 mars 2003 devant la commission d'examen

COMPOSITION DU JURY :

M. :	Claude	Labit	Président
MM. :	Ferran	Marques	Rapporteurs
	Luc	Vandendorpe	
MM. :	Michel	Barlaud	Examinateurs
	Pierre	Duhamel	
	Didier	Nicholson	
	Christine	Guillemot	

à Virginie

Remerciements

Je remercie chaleureusement tous le membres de mon jury : Claude LABIT, directeur de l'IRISA, qui m'a fait l'honneur de présider ce jury; Ferran MARQUES, professeur à l'Université Polytechnique de Catalogne (UPC) et Luc VANDENDORPE, professeur à l'Université Catholique de Louvain (UCL) qui ont bien voulu accepter la charge de rapporteur; Michel BARLAUD, professeur à l'ESSI, responsable de l'équipe CReATIVe au laboratoire I3S et Pierre DUHAMEL, directeur de recherches au CNRS, laboratoire L2S, qui ont bien voulu juger ce travail; Didier NICHOLSON, Thales Communications, responsable de la partie 11 de JPEG 2000 (JPWL), qui a bien voulu faire partie de ce jury en tant que membre invité.

J'aimerais remercier tout spécialement Christine GUILLEMOT, Directrice de recherche à l'INRIA et responsable du projet TEMICS, pour m'avoir permis de réaliser ces travaux de thèse, pour les avoir dirigés et pour m'avoir offert l'opportunité de les poursuivre.

Je tiens également à remercier tous ceux qui m'ont encouragé à me lancer dans cette thèse, parmi lesquels Jean-Luc Corre, Stéphane Pateux, qui a aussi participé à l'encadrement de la thèse, et bien sûr, Virginie.

Mon séjour à l'IRISA m'a permis et me permet encore de rencontrer de nombreuses personnes dont le contact est enrichissant, tant sur le plan professionnel que personnel. Je les salue amicalement. Enfin je salue plus particulièrement Jim et Jérôme, respectivement premier et troisième au classement de la saison 1999/2002.

Table des matières

Introduction **15**

1 Codage par descriptions multiples **19**
1.1 Introduction, principe 19
1.2 Etudes théoriques 21
1.2.1 Codage à deux descriptions 21
1.2.2 Codage à N descriptions 24
1.3 Techniques de codage par descriptions multiples, classification 25
1.3.1 Codage par descriptions multiples basé transformation 26
1.3.2 Codage MD basé quantification 37
1.3.3 Codage MD basé codes de canal 42
1.4 Codage vidéo à descriptions multiples 44
1.4.1 Codage vidéo à trois boucles de prédiction 45
1.4.2 Codage vidéo à deux boucles de prédiction 46
1.4.3 Autres techniques 47
1.4.4 Codage vidéo basé sous-bandes 3D 48
1.5 Conclusion 48

2 Codage progressif par descriptions multiples **51**
2.1 Introduction 51
2.2 Codage progressif à descriptions multiples 53
2.2.1 Définition 53
2.2.2 Transformée polyphase et quantification sélective 54
2.2.3 Quantification scalaire à descriptions multiples 55
2.3 Enrichissement de JPEG2000 par codage à descriptions multiples 63
2.3.1 Transformée polyphase et quantification sélective 64
2.3.2 MDUSQ 65
2.3.3 Allocation de redondance 70
2.4 Résultats expérimentaux 71
2.5 Conclusion 76

3 Décodage souple de codes arithmétiques 81
3.1 Introduction . . . 81
3.2 Notations . . . 83
3.3 Principe du codage arithmétique . . . 83
3.4 Modélisation . . . 85
3.4.1 Modèle à horloge symbole du codeur . . . 85
3.4.2 Modèle à horloge bit du décodeur . . . 86
3.5 Estimation . . . 90
3.5.1 Estimation à horloge symbole . . . 90
3.5.2 Estimation à horloge bit . . . 91
3.6 Elagage . . . 92
3.7 Codage arithmétique adaptatif . . . 93
3.8 Amélioration des performances par ajout de redondance . . . 94
3.8.1 Marqueurs de synchronisation . . . 94
3.8.2 Information adjacente . . . 97
3.8.3 Décodage turbo itératif . . . 97
3.8.4 Résultats expérimentaux . . . 99
3.9 Enrichissement de JPEG2000 par décodage souple de codes arithmétiques 104
3.9.1 Codage arithmétique contextuel dans EBCOT . . . 105
3.9.2 Application de l'algorithme de décodage souple . . . 106
3.9.3 Amélioration des performances par ajout de redondance . . . 106
3.9.4 Résultats expérimentaux . . . 107
3.10 Conclusion . . . 107

4 Décodage souple de codes quasi-arithmétiques 111
4.1 Introduction . . . 111
4.2 Notations . . . 112
4.3 Codage arithmétique à précision réduite . . . 113
4.3.1 Codage quasi-arithmétique . . . 113
4.3.2 Décodage quasi-arithmétique . . . 115
4.3.3 Approximation de la distribution de la source . . . 115
4.4 Modèle de source . . . 116
4.4.1 Conversion d'une source M-aire en source binaire . . . 117
4.4.2 Indexation pour la conversion . . . 118
4.5 Automates de codage et de décodage . . . 120
4.5.1 Modèle produit : source + codeur . . . 120
4.5.2 Modèle produit : source + décodeur . . . 122
4.5.3 Approximation de la distribution de la source . . . 124
4.5.4 Codage quasi-arithmétique adaptatif . . . 125
4.6 Estimation . . . 125
4.7 Amélioration des performances par ajout de redondance . . . 129
4.7.1 Marqueurs de synchronisation . . . 129
4.7.2 Information adjacente . . . 129
4.7.3 Décodage turbo itératif . . . 129

4.7.4 Résultats expérimentaux 131
4.8 Conclusion 139

5 Décodage souple de descriptions multiples 143
5.1 Introduction 143
5.2 Notations 144
5.3 Descriptions multiples et codes de Huffman 145
5.3.1 Modélisation 145
5.3.2 Estimation 146
5.3.3 Résultats expérimentaux 149
5.4 Descriptions multiples et codes quasi-arithmétiques 149
5.4.1 Modélisation 151
5.4.2 Estimation 152
5.4.3 Résultats expérimentaux 157
5.5 conclusion 160

Conclusion 165
5.6 Synthèse 165
5.7 perspectives 167

A JPEG 2000 169
A.1 Introduction 169
A.2 Codage EBCOT 170
A.2.1 Algorithme PCRD 171
A.2.2 Codage de blocs emboîté 172
A.2.3 Plans de bits partiels 173

B Réseaux bayésiens et inférence 175
B.1 introduction 175
B.2 Estimation MPM 176
B.2.1 Stratégies “organisées” 176
B.2.2 Stratégies “aveugles” 177
B.3 Estimation MAP 178
B.4 Information extrinsèque 179

C L’algorithme BCJR 181

Table des figures

1.1 Le problème de séparation de canaux 22
1.2 Organisation du codage à trois descriptions. 24
1.3 Exemple de schéma de compression classique. 25
1.4 Banc de filtres multi-ondelettes. 27
1.5 Reformulation du banc de filtres multi-ondelettes pour un signal mono-dimensionnel. 27
1.6 Banc de filtres à descriptions multiples. 28
1.7 Structure en cascade permettant de multiplier le nombre de descriptions obtenues par PCT. 31
1.8 Redondance introduite par interpolation spectrale. 33
1.9 Codage par transformée polyphase et quantification sélective. 35
1.10 (a) Quantification décalée, (b) représentation matricielle : les indices des quantificateurs grossiers correspondent aux colonnes et aux lignes. . . . 37
1.11 Quantification scalaire par descriptions multiples 38
1.12 Table d'index MDSQ linéaire modifiée [Vai93]. 39
1.13 (a) Exemple de réseau de points régulier pour la MDLVQ, (b) Exemple d'étiquetage optimal. 42
1.14 Construction de 4 descriptions à partir d'un train binaire progressif. . . 43
1.15 Codage vidéo par descriptions multiples basé sur 3 boucles de prédiction. Deux boucles sont représentées, la troisième se déduit par symétrie. . . . 46

2.1 Schéma bloc du codeur JPEG2000 52
2.2 Codage par descriptions multiples progressif par transformée polyphase : schéma bloc. 54
2.3 Codage par descriptions multiples progressif par transformée polyphase : entrelacement des plans de bits pour un rendement de 2/3. 55
2.4 Tables d'index MDSQ. 56
2.5 Comparaison des rapports distorsions centrale/latérale de la MDSQ avec la MDUSQ (2.6 bit/échantillon/description). 58
2.6 Forme générale d'une table d'index MDSQ. 60
2.7 Tables d'index MDSQ pour codage progressif. 63
2.8 Schéma bloc du codeur à descriptions multiples par transformée polyphase basé JPEG2000 64
2.9 Schéma bloc du codeur MDUSQ basé JPEG2000 65

2.10 Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 2$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$. 67
2.11 Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 4$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$. 67
2.12 Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 8$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$. 68
2.13 Courbes débit/distorsion pour le décodage central de Lena 512, comparaison des tables d'index linéaires modifiées et emboîtées ($d = 4$). 72
2.14 Courbes débit/distorsion pour le décodage latéral de Lena 512, comparaison des tables d'index linéaires modifiées et emboîtées ($d = 4$). 72
2.15 Courbes débit/distorsion pour le décodage central de Lena 512, comparaison de la transformée polyphase et de la MDUSQ. 73
2.16 Courbes débit/distorsion pour le décodage latéral de Lena 512, comparaison de la transformée polyphase et de la MDUSQ. 73
2.17 Compromis entre les distorsions centrale et latérale en fonction de la redondance dans le cas du décodage de Lena 512 à 0.5 bits/pixel, comparaison entre les deux codeurs proposés et deux codeurs de référence ([SRVN00] et [JO99]). 74
2.18 Compromis entre les distorsions centrale et latérale en fonction de la redondance dans le cas du décodage de Lena 512 à 0.25 bits/pixel, comparaison entre les deux codeurs proposés et deux codeurs de référence ([SRVN00] et [JO99]). 75
2.19 Codage JPEG2000-MDUSQ de Lena 512 avec $d = 4$ et 0.25 bits par pixel par description. 77
2.20 Codage JPEG2000-PT de Lena 512 avec 20% de redondance et 0.25 bits par pixel par description. 78

3.1 Principe du codage arithmétique. 84
3.2 Modèle à horloge symbole du codeur arithmétique ($N_k \geq N_{k-1}$). Les points blancs et noirs représentent respectivement les variables cachées et observées. 88
3.3 Modèle à horloge bit du décodeur arithmétique. Les points blancs et noirs représentent respectivement les variables cachées et observées. . . 89
3.4 Procédé d'estimation séquentielle à horloge symbole. 91
3.5 Procédé d'estimation séquentielle à horloge bit. 92
3.6 Modèle du codeur arithmétique avec des marqueurs de synchronisation binaires de longueur l ajoutés tous les f symboles. Les points gris sont les bits de synchronisation. Les points blancs et noirs sont respectivement les variables cachées et observées. 96

3.7 Représentation graphique à horloge bit des dépendances dans la chaîne de codage conjoint arithmétique-codage de canal. Les points blancs et noirs sont respectivement les variables cachées et observées. 98

3.8 SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman ($\rho = 0.1$, $K = 200$ symboles, 100 réalisations). 100

3.9 SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman ($\rho = 0.5$, $K = 200$ symboles, 100 réalisations). 100

3.10 SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman ($\rho = 0.9$, $K = 200$ symboles, 100 réalisations). 101

3.11 SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation ($\rho = 0.1$, $K = 200$ symboles, 100 réalisations). 101

3.12 SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation ($\rho = 0.5$, $K = 200$ symboles, 100 réalisations). 102

3.13 SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation ($\rho = 0.9$, $K = 200$ symboles, 100 réalisations). 102

3.14 SNR pour le décodage souple itératif de codes arithmétiques ($\rho = 0.1$, 100 réalisations) (a) comparé à la synchronisation souple ($K = 200$), (b) en fonction du nombre d'itérations pour différents paramètres d'élagage ($w = 2, 10, 50$, $K = 20$). 103

3.15 Illustration du compromis entre performance et complexité ($w = 2, 3, 5, 10$). 104

3.16 Décodage souple de Lena 512x512 codée par JPEG2000 à 0.25 bits par pixel. 108

3.17 Décodage souple de Lena 512x512 codée par JPEG2000 à 0.5 bits par pixel. 108

3.18 Impact des erreurs sur la qualité visuelle (0.5 bpp). 109

4.1 Représentation graphique sous la forme d'un a)-arbre, b)-automate stochastique, c)-treillis, du modèle binaire correspondant à une chaîne de Markov d'ordre 0, avec $M = 4$. Les nœuds noirs correspondent aux feuilles identifiées avec les racines des arbres suivants. Les nœuds blancs correspondent aux nœuds intermédiaires de la représentation binaire des symboles M-aires. 117

4.2 Représentation graphique sous la forme d'un a)-arbre, b)-automate stochastique, du modèle binaire correspondant à une chaîne de Markov d'ordre 1, avec $M = 4$. Les nœuds noirs correspondent aux feuilles identifiées avec les nœuds intermédiaires des arbres suivants. Les nœuds blancs correspondent aux autres nœuds intermédiaires. 118
4.3 Représentation graphique sous la forme d'un arbre du modèle binaire correspondant à une chaîne de Markov d'ordre 0, avec $M = 8$, a)-sans indexation particulière, b)-avec une indexation basé sur une permutation circulaire d'un symbole. Les probabilités de chaque branche ainsi que les probabilités des feuilles sont indiquées entre parenthèses. 119
4.4 Modèle produit source + codeur: a) états, sorties et transitions; b) Représentation en treillis. 121
4.5 Modèles de codeur quasi-arithmétique ($T = 4$): a) $0.63 \leq \mathbb{P}(MPS)$; b) $0.5 \leq \mathbb{P}(MPS) < 0.63$. 121
4.6 Structure de dépendances du modèle produit source + codeur ($N_k \geq N_{k-1}$). Les états blancs et noirs représentent respectivement les variables cachées et observées. 122
4.7 Modèle produit source + décodeur: a) états, sorties et transitions; b) Représentation en treillis. 123
4.8 Structure de dépendances du modèle produit source+décodeur. Les états blancs et noirs représentent respectivement les variables cachées et observées. 124
4.9 Exemple de treillis pour $K = 7$ et $N = 6$. 127
4.10 Procédé d'estimation en deux passes sur treillis. 128
4.11 Procédé d'estimation en deux passes sur treillis, construction du treillis séparée de l'estimation. 128
4.12 Représentation graphique à horloge bit des dépendances dans la chaîne de codage conjoint arithmétique-codage de canal. Les points blancs et noirs sont respectivement les variables cachées et observées. 130
4.13 SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques ($\rho = 0.5$, 200 symboles, 400 réalisations). . . . 132
4.14 SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques ($\rho = 0.9$, 200 symboles, 400 réalisations). . . . 132
4.15 SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques, avec synchronisation souple ($\rho = 0.5$, 200 symboles, 400 réalisations). 133
4.16 SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques, avec synchronisation souple ($\rho = 0.9$, 200 symboles, 400 réalisations). 134

4.17 SER et SNR pour le décodage souple de codes quasi-arithmétiques (a) avec synchronisation souple et (b) avec information adjacente ($\rho = 0.5$, 200 symboles, 400 réalisations). 134
4.18 SER et SNR pour (a) le décodage souple de codes quasi-arithmétiques et (b) le décodage turbo itératif de codes quasi-arithmétiques ($\rho = 0.5$, 200 symboles, 400 réalisations). 135
4.19 SER et SNR pour (a) le décodage souple de codes quasi-arithmétiques et (b) le décodage turbo itératif de codes quasi-arithmétiques ($\rho = 0.9$, 200 symboles, 400 réalisations). 136
4.20 SER et SNR pour le décodage souple de codes quasi-arithmétiques avec (a) $T = 4$ sans redondance, (b) $T = 8$ sans redondance et (c) $T = 8$ avec des marqueurs de synchronisation ($\rho = 0.5$, 100 symboles, 400 réalisations). 137
4.21 SER et SNR pour le décodage souple de codes quasi-arithmétiques (a) sans indexation et (b) avec indexation ($\rho = 0.1$, 200 symboles, 400 réalisations). 137
4.22 Nombre moyen d'erreurs symboles par index dans la séquence ($\rho = 0.5$, 50 symboles, 500 réalisations). 139

5.1 Structure de dépendances du modèle source de Markov + MDUSQ + codeur de Huffman. Les états blancs et noirs représentent respectivement les variables cachées et observées. 147
5.2 Décodage souple de MDUSQ et de codes de Huffman. 148
5.3 SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement 5/6). $\rho = 0.1$. 150
5.4 SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement 5/6). $\rho = 0.5$. 151
5.5 SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement 5/6). $\rho = 0.9$. 152
5.6 Schéma bloc d'un codeur quasi-arithmétique à deux descriptions. 153
5.7 Structure de dépendances du modèle source de Markov M-aire + MDUSQ + conversion vers source binaire + codeur quasi-arithmétique, pour une seule description (la deuxième description est symétrique). On a $k = l \times q$. Les états blancs et noirs représentent respectivement les variables cachées et observées. 153
5.8 Estimation simplifiée de codes quasi-arithmétiques à deux descriptions. . 157
5.9 SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques ($\rho = 0.1$). 158
5.10 SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques ($\rho = 0.5$). 159
5.11 SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques ($\rho = 0.9$). 159

5.12 SNR pour le décodage souple de codes quasi-arithmétiques avec et sans MDUSQ, pour des taux d'effacement respectivement de 10%, 5% et 1% ($\rho = 0.1$). 160
5.13 SNR pour le décodage souple de codes quasi-arithmétiques avec et sans MDUSQ, avec et sans effacements ($\rho = 0.1$). 161
5.14 Proposition de codeur robuste basé MDUSQ et codage quasi-arithmétique. 162
5.15 Proposition de décodeur robuste basé MDUSQ et codage quasi-arithmétique. 163

A.1 Organisation des couches de qualité dans EBCOT. Par exemple, le bloc 7 ne contribue pas à la couche 1. 170
A.2 Briques principales de l'algorithme EBCOT. 171
A.3 Contextes formés pour le codage arithmétique des primitives 173
A.4 Propriétés débit-distorsion (a) du codage par plans de bits régulier et (b) du codage par plans de bits partiels. 174
A.5 Ordre des passes dans le train binaire EBCOT final d'un bloc B_i. 174

B.1 Processus de Markov X et mesure associée Y (a) sur les variables de X et (b) sur les transitions de X. 175

C.1 Schéma du système de transmission. 181
C.2 Exemple de source de Markov à trois états. (a) graphe des transitions entre états, (b) treillis correspondant. 182

Glossaire

AC	Arithmetic Coding
ARQ	Automatic Repeat reQuest
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BCJR	du nom des auteurs de l'algorithme : Bahl, Cocke, Jelinek et Raviv
BPSK	BiPhase Shift Keying
EBCOT	Embedded Block Coding with Optimized Truncation
CABAC	Context Adaptive Binary Arithmetic Coding
CC	Code Convolutif
DCT	Discrete Cosinus Transform
DFT	Discrete Fourier Transform
DPCM	Differential Pulse Code Modulation
EEP	Equal Error Protection
EQM	Erreur Quadratique Moyenne
FEC	Forward Error Correction
ITU	International Telecommunications Union
LOT	Lapped Orthogonal Transform
LUT	Look Up Table
MAP	Maximum A Posteriori
MDC	Multiple Description Coding
MDSQ	Multiple Description Scalar Quantization
MDTC	Multiple Description Transform Coding
MDUSQ	Multiple Description Uniform Scalar Quantization
MDVC	Multiple Description Video Coding
MDVQ	Multiple Description Vector Quantization
MDLVQ	Multiple Description Lattice Vector Quantization
MPM	Maximum of Posterior Marginals
MWT	Multi-Wavelet Transform
PCT	Pairwise Correlating Transform
PGZ	Peterson Gorenstein Zierler
POCS	Projection Onto Convex Sets
PSEC	Projection Sur Ensemble Convexe (voir POCS)
PSNR	Peak Signal to Noise Ratio

QAC	Quasi-Arithmetic Coding
RCPC	Rate Compatible Punctured Code
RS	Reed-Solomon
RVLC	Reversible Variable Length Code
SER	Symbol Error Rate
SNR	Signal to Noise Ratio
SPIHT	Set Partitioning In Hierarchical Tree
TCM	Trellis Coded Modulation
TCP	Transmission Control Protocol
TCQ	Trellis Coded Quantization
UEP	Unequal Error Protection
ULP	Unequal Loss Protection
VLC	Variable Length Code
VM	Verification Model

Introduction

Contexte de l'étude et problématique. Cette thèse s'inscrit dans le contexte de la transmission de données image et vidéo sur des réseaux de paquets hétérogènes aux caractéristiques variant dans le temps. Depuis le développement d'internet, les besoins en communication multimédia se sont multipliés. Des applications telles que la diffusion (*streaming*) et la visioconférence, mais aussi la télémédecine, la vidéo surveillance et bien d'autres encore nécessitent la transmission de telles données avec des contraintes de qualité, de fiabilité et de délai. Une première difficulté dans la conception de schémas de communications qui respectent ces contraintes réside dans l'aspect hétérogène à la fois du canal de transmission et des récepteurs. Il est concevable par exemple de participer à une visioconférence aussi bien à partir d'un ordinateur personnel qu'à partir d'un téléphone portable. Dans ce cas, il y a une différence énorme entre les capacités de ces émetteurs/récepteurs à traiter la vidéo. Les canaux de transmission, dans le cas d'internet, peuvent être composés de portions filaires et non-filaires dont les capacités varient. Le protocole de transmission le plus utilisé sur l'internet filaire est TCP. Ce protocole peut être sujet à des congestions, auquel cas des paquets seront éliminés aléatoirement. Le protocole TCP s'appuie sur un mécanisme d'acquittement et de demande de retransmission des paquets perdus (ARQ) pour garantir la fiabilité de la communication. Cependant, ce mécanisme ne permet pas de respecter les contraintes de délai inhérentes aux applications de type streaming, visioconférence ou télémédecine. Lorsque l'internet est accédé par des liens sans fil, comme par exemple à partir d'un téléphone portable, les données sont en plus victimes des perturbations propre à ce type de communication, telles que les évanouissements ou les rafales d'erreurs. Traditionnellement, le codage de canal est utilisé pour protéger les transmissions sans fil. Au problème de l'hétérogénéité du réseau vient s'ajouter celui de la non-stationnarité de ses caractéristiques. Des schémas de transmission vidéo qui utilisent des moyens d'observation de l'état du réseau pour répartir de manière optimale le débit disponible entre le codage de source et le codage de canal ont été proposés. Cette capacité d'adaptation constitue une caractéristique importante du schéma de codage. Les techniques de codage progressif ou "scalable" permettent d'adapter aisément le débit du codage de source. Cependant, elles introduisent une notion de priorité dans les données codées. Des schémas de protection inégale ont été développés spécifiquement pour cette situation.

Dans cette thèse, nous cherchons à traiter le problème général de la conception de schémas de transmission d'information multimédia sur des réseaux hétérogènes et

variant dans le temps. Les solutions proposées ont pour but de permettre

- la robustesse aux pertes de paquets (effacements),
- la robustesse aux perturbations des transmissions sans fil (erreurs)
- le respect de contraintes de délai,
- l'adaptation aux caractéristiques de perte, d'erreur et de débit du canal
- et l'adaptation aux caractéristiques des émetteurs/récepteurs.

L'étude du codage par descriptions multiples a été choisie comme point de départ de ce travail.

Codage conjoint source canal. L'utilisation du codage de canal indépendamment du codage de source est liée au théorème de séparation source/canal énoncé par Shannon [Sha48]. Selon ce théorème, des performances optimales peuvent ainsi être obtenues. Cependant, en plus de la difficulté pratique à atteindre les limites citées par ce théorème, les hypothèses de celui-ci sont rarement vérifiées. Ce constat est à la base du développement des techniques de *codage conjoint source/canal*. Ce terme désigne l'ensemble des techniques de transmission où le codage de source et de canal sont conçus conjointement en fonction de la connaissance des caractéristiques du canal. Les techniques d'optimisation de la répartition du débit disponible ou de protection inégale des données mentionnées ci-dessus sont des exemples de codage conjoint source/canal. D'autres approches consistent à laisser volontairement de la redondance dans les données pendant l'étape de codage de source, de façon à ce que celle-ci puisse être exploitée au décodage. Le codage par descriptions multiples, que nous étudions dans cette thèse se classe également dans les techniques de codage conjoint source/canal. En effet, le codage par descriptions multiples trouve son origine dans l'idée de répartir des données entre plusieurs canaux distincts, de façon à ce qu'en cas de panne d'un canal, la totalité de ces données ne soit pas perdue. Le codeur de source doit alors permettre la répartition des données autrement que par simple duplication.

Codage par description multiples. Le codage par descriptions multiples consiste à créer plusieurs représentations distinctes mais corrélées d'une source qui ont la propriété de se raffiner mutuellement. Ces représentations sont appelées des descriptions. La réception d'une seule description quelle qu'elle soit doit permettre une reconstruction de la source avec un niveau de qualité acceptable. Chaque description supplémentaire reçue doit permettre d'améliorer la qualité de reconstruction. La qualité optimale est obtenue quand toutes les descriptions sont reçues. Le cas particulier du codage à deux descriptions a fait l'objet d'études approfondies, aussi bien théoriquement que pratiquement. Par construction, le codage par descriptions multiples est bien adapté à la transmission sur plusieurs canaux indépendants ou sur un canal à effacements sans mémoire. Il a également l'avantage de favoriser le respect des contraintes de délai, puisqu'il n'y a pas besoin d'attendre que la totalité des descriptions soient reçues pour pouvoir décoder les données.

Organisation du document et contributions. Ce document est organisé comme suit. Tout d'abord le chapitre 1 présente un état de l'art du codage par descriptions multiples, découpé en trois parties. Dans une première partie, des résultats théoriques sont présentés sur les performances du codage à deux descriptions et à N descriptions. Ensuite, dans une seconde partie, un état de l'art des techniques de codage par descriptions multiples est proposé. Ces techniques sont classées suivant la position qu'elles occupent dans une chaîne de codage générique. Trois catégories sont distinguées : les techniques basées transformation, les techniques basées quantification et les techniques basées codage de canal. Enfin, dans la dernière partie, l'application du codage par descriptions multiples à la vidéo est étudiée. Des exemples de codeurs vidéo à deux descriptions sont présentés.

L'état de l'art permet de mettre en avant les avantages et inconvénients de chaque technique de codage par descriptions multiples. La quantification scalaire à descriptions multiples (MDSQ) est mise en avant et utilisée dans le reste du document. Dans le chapitre 2, l'application de cette dernière au codage d'image fixe progressif est considérée. La caractéristique de progressivité est primordiale dans la conception de codeurs qui permettent d'adapter aisément le débit des données à transmettre. Notre contribution réside dans la création d'un codeur à deux descriptions progressives, ce qui signifie que chaque description peut être transmise et décodée partiellement, en fonction du débit disponible sur le canal ou sur chacun des canaux, suivant le contexte. Les deux descriptions peuvent contribuer inégalement à la reconstruction. L'étude montre que le point clé de la MDSQ, la table d'index, ne permet pas un codage progressif efficace. Nous proposons un algorithme qui permet la création de tables d'index adaptées au codage progressif des descriptions. Finalement, la méthode est intégrée dans un codeur d'image fixe JPEG2000. Une comparaison avec d'autres codeurs d'image fixe à deux descriptions montre la compétitivité de notre approche en terme de rapport débit-distorsion. Notre codeur, en plus de la progressivité, hérite de toutes les fonctionnalités de JPEG2000.

Le codeur d'image à deux descriptions présente une robustesse intéressante face aux effacements. Il n'est en revanche pas conçu pour résister aux erreurs. Une étude de ce codeur, et plus généralement de la structure globale d'une chaîne de compression classique, met rapidement en évidence un point faible : le codage entropique. Tous les systèmes de compression actuels incluent une étape de codage entropique. Pour les systèmes les plus récents, tels que JPEG2000, il s'agit du codage arithmétique. Ces codes sont particulièrement fragiles face aux erreurs. En effet, une seule erreur bit suffit à désynchroniser un décodeur arithmétique. Tous les symboles qui suivent l'erreur sont alors erronés. Lorsque les codes de canal utilisés par le protocole réseau ne permettent pas de garantir une erreur résiduelle nulle, comme c'est le cas pour les transmissions sans fil, même un très faible taux d'erreur peut entraîner des résultats catastrophiques. Il existe de nombreux travaux qui traitent de la synchronisation des codes de Huffman. Peu de travaux, en revanche, considèrent les codes arithmétiques.

Dans le chapitre 3, nous considérons ce problème en détail. Une modélisation basée sur les réseaux bayésiens permet de mettre en évidence les dépendances et les contraintes entre les différentes variables de la chaîne de codage. Un algorithme de décodage souple est alors développé. La fiabilité de l'estimation peut être renforcée en ajoutant de la

redondance sous forme de marqueurs de synchronisation ou d'information adjacente. Cette redondance est exploitée par l'algorithme d'estimation et traite spécifiquement le problème de la synchronisation des codes arithmétiques. Un schéma itératif inspiré des turbo codes en série est également présenté. Les performances de ce dernier sont cependant limitées. En effet, de part la structure même du codage arithmétique, l'algorithme d'estimation souffre d'une complexité exponentielle. Des techniques d'élagage adaptées permettent de contrôler cette complexité, mais rendent l'estimation sous-optimale. De bonnes performances sont tout de même obtenues, hormis dans le cas du schéma itératif. L'algorithme de décodage souple est finalement intégré dans un codeur d'image fixe JPEG2000. Les résultats expérimentaux montrent des gains de PSNR qui peuvent dépasser les 15 dB.

La présence de l'élagage dans l'algorithme de décodage souple de codes arithmétiques rend celui-ci sous-optimal. Comme ce problème est inhérent à la structure même des codes arithmétiques, nous remettons ces derniers en question. Dans le chapitre 4, des codes arithmétiques simplifiés appelés codes quasi-arithmétiques sont considérés. De nouveau, le formalisme des réseaux bayésiens est utilisé pour modéliser la chaîne de codage. Les propriétés du codage quasi-arithmétique associées à une modélisation appropriée de la source permettent de représenter le codeur et le décodeur sous forme d'automates à nombre d'états finis. Un nouvel algorithme de décodage souple, optimal cette fois, est développé. Les marqueurs de synchronisation et l'information adjacente peuvent également être exploités par cet algorithme. Un nouveau schéma de décodage itératif est développé. Les itérations apportent cette fois un gain significatif.

Le codage par descriptions multiples, qui permet de concevoir des codeurs robustes aux effacements et le décodage souple de codes arithmétiques, qui permet de concevoir des codeurs robustes aux erreurs sont étudiés séparément et répondent chacun à une partie seulement des objectifs de la thèse cités ci-dessus. Dans le chapitre 5, ces deux approches sont fusionnées dans le but de concevoir un codeur robuste à la fois aux erreurs et aux effacements. La difficulté dans cette fusion consiste à faire collaborer les deux approches, plutôt que de les concaténer simplement. Là encore, le formalisme des réseaux bayésien nous aide à modéliser l'ensemble de la chaîne de codage. Ainsi des algorithmes de décodage souple qui exploitent la redondance introduite par le codage à descriptions multiples peuvent être conçus. Deux cas sont étudiés : la MDSQ associée aux codes de Huffman et la MDSQ associée au codes quasi-arithmétiques. Finalement, ce dernier cas est utilisé pour proposer un schéma général de codage robuste aux effacements et aux erreurs et adaptable aux caractéristiques du canal et des émetteurs/récepteurs.

En conclusion, une synthèse des approches proposées et des résultats obtenus est faite, et des perspectives sont données.

Chapitre 1

Codage par descriptions multiples

1.1 Introduction, principe

Actuellement, de plus en plus d'applications nécessitent la transmission de données audio ou vidéo à travers des réseaux de paquets. Des contraintes de débit et de délai doivent alors être respectées. Le principe du codage progressif ou *scalable* [EC91] permet de faire face au problème de l'adaptation du débit à une bande passante non stationnaire. La qualité des données reçues dépend alors du nombre de *couches* qui arrivent à destination. Ces couches sont hiérarchisées. C'est à dire qu'une couche N ne peut pas être exploitée sans les couches précédentes $1, 2, ..., N-1$. Les réseaux de transmission utilisés actuellement, tels que l'internet, peuvent être victimes de congestions, ce qui entraîne la suppression de paquets. Les techniques de retransmission automatique (*automatic repeat request, ARQ*) permettent de récupérer ces paquets perdus, à condition de ne pas avoir de contrainte de délai à respecter. Dans ce dernier cas, le codage progressif constitue une solution viable uniquement si il existe une notion de priorité dans le protocole réseau utilisé. Les priorités sont alors réglées de façon à ce que toute couche N soit prioritaire par rapport à la couche $N+1$. Cependant, dans les protocoles utilisés couramment, tel que TCP sur internet, la notion de priorité n'existe pas. Les pertes de paquets sont aléatoires.

Pour pallier ce problème, l'idéal serait d'avoir un système de codage qui répartit les données dans des paquets et dont la qualité de reconstruction dépend uniquement du nombre de paquets reçus, et non pas de quels paquets sont reçus. On dit dans ce cas que les paquets se raffinent mutuellement et on parle de *codage par descriptions multiples*.

Le codage par descriptions multiples trouve son origine dans les laboratoires Bell, dans les années 1970 [Goy01]. Le problème qui se posait à l'époque était d'assurer la fiabilité des communications téléphoniques. La transmission sur deux lignes distinctes pouvait permettre de maintenir la continuité d'une conversation en cas de panne d'une des deux lignes. Il devenait intéressant dans ce cas de ne pas dupliquer l'information,

mais plutôt de la répartir entre les deux lignes de manière à ce que la qualité soit maximale lorsque les deux lignes fonctionnent et dégradée mais suffisante pour poursuivre la conversation en cas de panne d'une des deux lignes. Cela a donné une première définition du codage par descriptions multiples. Par la suite cette idée a été délaissée. Seuls des résultats théoriques ont été publiés sur le sujet au début des années 1980 [WWZ80] [Oza80] [GC82]. Récemment, le codage par descriptions multiples a connu un regain d'intérêt. De nombreuses techniques permettant de mettre en oeuvre ce principe ont été proposées ([Vai93], [Hem96], [OWVR97], ...). L'application principale du codage par descriptions multiple est la transmission de données soumises à des contraintes de délai, telles que l'audio et la vidéo sur des réseaux de type internet, filaires ou non filaires.

Définition : soit une source d'information X. Un *codeur à N descriptions* permet de générer N représentation $X_1, X_2, \ldots, X_N$ différentes mais corrélées de X appelées descriptions et telles que si on a deux ensembles quelconques d'indices $I \subset \{1, 2, \ldots, N\}$ et $J \subset \{1, 2, \ldots, N\}$, alors $|I| > |J|$ implique $D(X, \hat{X}_I) < D(X, \hat{X}_J)$, avec $\hat{X}_I$ la source X reconstruite par le décodeur à partir des descriptions indicées par I et $D(\cdot, \cdot)$ une mesure de distorsion.

Le codage par descriptions multiples est souvent étudié dans le cas particulier de deux descriptions. Dans ce cas, la source X est représentée par deux descriptions X_1 et X_2. Au décodage, trois configurations sont possibles :

- seule X_1 est reçue. X est reconstruite avec la distorsion $D_1 = D(X, \hat{X}_1)$.
- Seule X_2 est reçue. X est reconstruite avec la distorsion $D_2 = D(X, \hat{X}_2)$.
- Les deux descriptions X_1 et X_2 sont reçues. X est reconstruite avec la distorsion $D_0 = D(X, \hat{X}_{1,2})$, telle que $D_0 < D_1$ et $D_0 < D_2$.

Définition : En codage à deux descriptions, lorsque les deux descriptions sont reçues, on parle de *décodage central*.

Définition : En codage à deux descriptions, lorsqu'une seule des deux descriptions est reçue, on parle de *décodage latéral*.

Définition : En codage à deux descriptions, on dit que les deux descriptions sont *équilibrées* lorsque $D_1 = D_2$ et *déséquilibrées* lorsque $D_1 \neq D_2$.

Cette dernière définition peut s'étendre au cas du codage à N descriptions. On remarque que le codage progressif est un cas particulier de codage à descriptions multiples où les descriptions sont déséquilibrées [EC91].

Dans la pratique, le codage par descriptions multiples est généralement utilisé sous les hypothèses suivantes :

- Les pertes ne sont pas corrélées. Cette hypothèse est vérifiée dans le cas de la transmission sur des canaux distincts non corrélés ou bien dans le cas de la transmission sur un canal sans mémoire.
- L'erreur résiduelle est nulle. Si le canal est sujet à des erreurs, on suppose que la partie codage de canal du système de transmission suffit à corriger ces erreurs. On est alors dans un cas idéal où les descriptions sont soit reçues sans erreur, soit perdues complètement.

Le fait de pouvoir reconstruire la source avec un sous-ensemble, voire une seule des descriptions impose que celles-ci soient corrélées. Toute la difficulté dans la conception d'un système de codage à descriptions multiples réside dans le réglage de cette corrélation, de manière à obtenir une qualité de reconstruction maximale lorsque toutes les descriptions sont reçues, tout en garantissant une qualité suffisante lorsqu'une seule des descriptions est reçue. D'une manière générale, une faible corrélation permet d'obtenir une meilleure qualité lorsque toutes les descriptions sont reçues, au détriment de la qualité obtenue lorsque seul un sous-ensemble des descriptions est reçu. La redondance introduite par le codage par descriptions multiples par rapport au codage classique est dans ce cas moins importante. Cette redondance est rarement mesurée explicitement. Quand nous avons besoin de comparer le codage par descriptions multiples au codage classique à simple description, nous mesurons la redondance de la manière suivante. Soit R_n, le débit disponible pour coder la source X. Soit D_0 la distorsion obtenue quand X est codée en N descriptions au débit total R_n et décodée à partir des N descriptions. Soit R_k le débit nécessaire pour coder X avec un codeur de source classique de manière à obtenir la distorsion D_0. Par analogie au codage de canal, nous dirons que le *rendement équivalent* du codeur à descriptions multiples est R_k/R_n.

Dans ce chapitre, nous présentons tout d'abord une partie des études théoriques qui ont été effectuées sur le codage par descriptions multiples. Ensuite, nous faisons un état de l'art des techniques existantes. Puis nous nous intéressons au cas particulier du codage vidéo à descriptions multiples, avant de conclure. Un état de l'art du codage par descriptions multiples a déjà été publié par Goyal [Goy01].

1.2 Etudes théoriques

1.2.1 Codage à deux descriptions

Dans ce paragraphe nous nous intéressons au problème de communication posé par la transmission d'une source gaussienne de variance unitaire vers trois récepteurs. Deux canaux sont disponibles, chacun menant à un récepteur différent. Un troisième récepteur peut recevoir les deux canaux. Ozarow obtient dans [Oza80] une expression des distorsions atteignables simultanément par chacun des récepteurs (au sens de l'erreur quadratique moyenne, EQM). En s'appuyant sur cette relation il peut démontrer quelles sont les limites de qualité de reconstruction atteignables lorsque les deux liens fonctionnent et lorsqu'une dégradation gracieuse est souhaitée dans l'éventualité d'un dysfonctionnement de l'un des liens.

Soit une source générant un message $\mathbf{X}$ constitué de N lettres X_k prises dans un alphabet χ. On suppose les X_k indépendantes et identiquement distribuées avec une densité de probabilité $p(x)$. Soient $f_1(\mathbf{X})$ et $f_2(\mathbf{X})$ les sorties de deux codeurs prenant $\mathbf{X}$ en entrée.

Soient trois récepteurs devant estimer $\mathbf{X}$ en utilisant respectivement f_1 seule, f_2 seule, et f_1 et f_2. Les trois estimations $\hat{\mathbf{X}}_1$, $\hat{\mathbf{X}}_2$ et $\hat{\mathbf{X}}_0$ sont des messages de N lettres prises dans des alphabets χ_1, χ_2 et χ_0. En général les trois alphabets ne coïncideront ni entre eux, ni avec l'alphabet χ de départ. Les distorsions D_1, D_2 et D_0 pourront être

calculées par chaque récepteur avec:

$$D_i = \frac{1}{N}\sum_{k=1}^{N} E\left[\delta_i(X_k, \hat{\mathbf{X}}_{ik})\right] \tag{1.1}$$

où $\delta_i(.,.)$ est une fonction à valeurs réelles non négatives.

Shannon montre dans [Sha59] que la fonction de débit-distorsion peut s'écrire:

$$R(D) = \inf_{P_D} I(\mathbf{X}, \hat{\mathbf{X}}) \tag{1.2}$$

où $P_D = \{p(\hat{x}|x) : E[\delta(\mathbf{X}, \hat{\mathbf{X}})] \leq D\}$ et $I(\mathbf{X}, \hat{\mathbf{X}})$ est l'information mutuelle entre $\mathbf{X}$ et $\hat{\mathbf{X}}$ [Pap91]. On peut montrer que pour tout D (pour lequel P_D n'est pas nul) et tout $\epsilon > 0$ il existe un bloc de longueur N et un code ayant au plus $e^{R(D)N}$ mots tels que $\hat{\mathbf{X}}$ satisfait

$$\frac{1}{N}\sum_{k=1}^{N} E\left[\delta_i(X_k, \hat{\mathbf{X}}_{ik})\right] < D + \epsilon \tag{1.3}$$

Inversement, pour tout code de débit inférieur à $R(D)$, la distorsion ne peut être inférieure à D. Cette dernière remarque peut aussi s'exprimer de la manière suivante si $I(\mathbf{X}, \hat{\mathbf{X}}) \leq NR$,

$$\frac{1}{N}\sum_{k=1}^{N} E\left[\delta_i(X_k, \hat{\mathbf{X}}_{ik})\right] \geq D^* \tag{1.4}$$

ou $R(D^*) = R$.

Si l'on se réfère au schéma de la figure (1.1), le problème est de caractériser les quin-

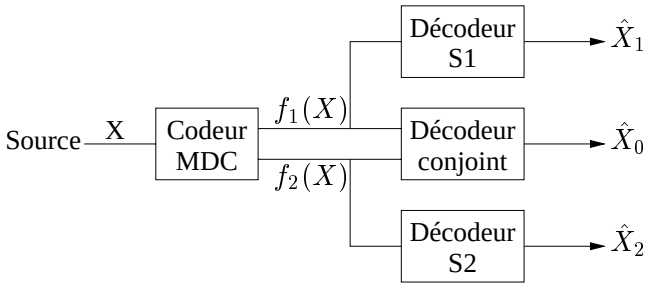


FIG. 1.1 – *Le problème de séparation de canaux*

tuples (R_1,R_2,D_1,D_2,D_0). La solution de ce problème n'a pas encore été trouvée dans un cas général. Toutefois, Ozarow l'a obtenu dans le cas particulier d'une source gaussienne avec une distorsion sous forme d'erreur quadratique. Dans ce cas

$$\delta_i(x, \hat{x}) = (x - \hat{x})^2, i = 1, 2, 0 \tag{1.5}$$

La fonction débit distorsion pour une telle source est donnée par

$$R(D) = \frac{1}{2} \log \frac{\sigma_x^2}{D} \tag{1.6}$$

où σ_x^2 est la variance de $\mathbf{X}$, supposée égale à 1 ici. On peut noter que $R(D)$ donne l'information mutuelle minimale nécessaire pour reproduire la source $\mathbf{X}$ avec la distorsion moyenne D. Cette relation est inversible:

$$D(R) = e^{-2R} \tag{1.7}$$

A l'inverse de (1.6), nous avons ici la distorsion moyenne minimale atteignable en représentant le vecteur de taille N $\mathbf{X}$ par $\hat{\mathbf{X}}$, quand l'information mutuelle entre $\mathbf{X}$ et $\hat{\mathbf{X}}$ est inférieure ou égale à NR.

Au final en remarquant que $\hat{\mathbf{X}}_i$ est une fonction de $f_i(\mathbf{X})$ pour $i = 1, 2, 0$ on aboutit à

$$\begin{gathered} I(\mathbf{X}; \hat{\mathbf{X}}_1) \leq NR_1 \\ I(\mathbf{X}; \hat{\mathbf{X}}_2) \leq NR_2 \\ I(\mathbf{X}; \hat{\mathbf{X}}_0) \leq N(R_1 + R_2) \end{gathered}$$

ce qui nous permet de conclure avec la relation (1.4) que:

$$\begin{gathered} D_1 \geq exp(-2R_1) \\ D_2 \geq exp(-2R_2) \\ D_0 \geq exp[-2(R_1 + R_2)] \end{gathered} \tag{1.8}$$

Dans un problème avec un seul destinataire, on peut montrer avec la relation (1.3) que la courbe distorsion-débit peut être approchée avec une précision arbitrairement petite, avec des tailles de blocs N suffisamment grandes. Appliquée à notre cas, cette remarque pourrait permettre de remplacer les inégalités des relations (1.8) par des égalités approximatives. Ozarow démontre dans [Oza80] qu'il est impossible d'atteindre de telles performances et que les quintuples atteignables doivent pour celà plutôt satisfaire les relations suivantes:

$$\begin{gathered} D_1 \geq exp(-2R_1) \\ D_2 \geq exp(-2R_2) \\ D_0 \geq exp[-2(R_1 + R_2)] \frac{1}{1-(\sqrt{\pi}-\sqrt{\Delta})^2} \end{gathered} \tag{1.9}$$

où $\pi = (1 - D_1)(1 - D_2)$ et $\Delta = D_1 D_2 - exp[-2(R_1 + R_2)]$. Avec ces dernières relations on peut maintenant démontrer quelles sont les limites de qualité de reconstruction atteignables lorsque les deux liens fonctionnent et lorsqu'une dégradation gracieuse est souhaitée dans l'éventualité d'un dysfonctionnement de l'un des liens. On s'aperçoit ainsi que si les performances en présence des deux descriptions approchent l'optimum théorique ($D_0 \cong exp[-2(R_1 + R_2)]$) alors la distorsion moyenne en présence de dysfonctionnements deviendra grande. Ainsi on peut espérer une diminution par deux de la distorsion lors de la réception de deux canaux par rapport à la réception d'un seul canal

si les deux descriptions prises individuellement n'ont pas de fortes contraintes en terme de qualité [GC82]. D'un autre coté, si des performances quasi-idéales sont souhaitées en présence de dysfonctionnements, la distorsion en présence des deux descriptions sera loin de son optimum. En d'autres termes, si les contraintes de distorsion sur les deux descriptions sont sévères, deux descriptions ne donneront rien de mieux qu'une seule puisque en fait les deux descriptions seront identiques.

1.2.2 Codage à N descriptions

Dans un premier temps, l'étude théorique du codage par descriptions multiples a été limitée au cas particulier du codage à deux descriptions. Récemment, Puri, Pradhan et Ramchandran [PPR02a] [PPR02b] sont parvenus à caractériser les performances du problème du codage à N descriptions. Plus précisément, ils présentent une "région débit/distorsion" atteignable par un système de codage à N descriptions. Ce résultat est une généralisation de la région obtenue par El Gamal et Cover [GC82] pour $N = 2$. Il a été développé en lien avec la technique de codage par descriptions multiples basée sur la protection inégale aux pertes (section 1.3.3). De ce fait, il nécessite la condition de raffinement successif de l'information [EC91]. Nous énonçons ce résultat pour $N = 3$. Il peut être généralisé à $N > 3$.

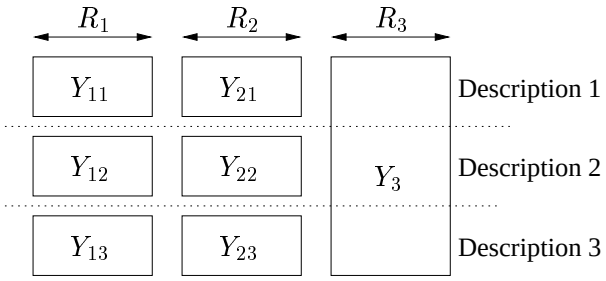


FIG. 1.2 – *Organisation du codage à trois descriptions.*

Soit X une source de fonction densité de probabilités $q(x)$. Soient Y_{11}, Y_{12}, Y_{13}, Y_{21}, Y_{22}, Y_{23} et Y_3 les variables aléatoires représentées sur la figure 1.2. Y_{11}, Y_{12} et Y_{13} représentent une "première partie" de X associée à un code correcteur $(1, 3)$. Il suffit de recevoir une description pour exploiter cette information. Y_{21}, Y_{22} et Y_{23} représentent une "seconde partie" de X associée à un code correcteur $(2, 3)$. Deux descriptions sont nécessaires pour exploiter cette information. Enfin, Y_3 représente une "dernière partie" de X, qui est exploitée uniquement si les trois descriptions sont reçues. Y_3 prend sa valeur dans l'alphabet $\mathcal{Y}_3$ et Y_{ij} dans l'alphabet $\mathcal{Y}_{ij}$, avec $i \in I$, $I = \{1, 2\}$ et $j \in J$, $J = \{1, 2, 3\}$. Le débit total par description R est partitionné en $R_1 + R_2 + R_3$. Y_{11}, Y_{12} et Y_{13} sont réparties entre les trois descriptions avec un débit R_1 par description. Y_{21}, Y_{22} et Y_{23} sont réparties entre les trois descriptions avec un débit R_2 par description. Enfin Y_3 est répartie entre les trois descriptions avec un débit R_3 par description.

Théorème : Une région de débits atteignables pour un triplet de distorsions $(D_1, D_2,$

D_3) donné est donnée par $R = R_1 + R_2 + R_3$, où

$$R_1 > H(Y_{11}) - \frac{1}{3}H(Y_{1J}|X),$$

$$R_2 > \frac{1}{2}H(Y_{2I}|Y_{1I}) - \frac{1}{3}H(Y_{2J}|X) \text{ et}$$

$$R_3 > \frac{1}{3}I(X;Y_3|Y_{IJ})$$

pour une fonction densité de probabilités $p(x, y_{IJ}, y_3) = q(x)p(y_{1J}|x)p(y_{2J}|x)p(y_3|x, y_{IJ})$, où $Y_{1J} \to X \to Y_{2J}$ est une chaîne de Markov et où on a un ensemble de fonctions de décodage

$$g_i^1 : \mathcal{Y}_{1i} \to \hat{\mathcal{X}},\ i \in J,$$

$$g_S^2 : \bigotimes_{k\in I, i\in S} \mathcal{Y}_{ki} \to \hat{\mathcal{X}},\ \forall S \subset J,\ |S| = 2 \text{ et}$$

$$g^3 : \mathcal{Y}_3 \times \bigotimes_{k\in I, j\in J} \mathcal{Y}_{kj} \to \hat{\mathcal{X}},$$

telles que

$$E[d(X, g_i^1(Y_{1i}))] \leq D_1,\ \forall i \in J,$$

$$E[d(X, g_S^2(Y_{IS}))] \leq D_2,\ \forall S \subset J,\ |S| = 2 \text{ et}$$

$$E[d(X, g^3(Y_{IJ}, Y_3))] \leq D_3.$$

1.3 Techniques de codage par descriptions multiples, classification

Nous proposons à présent un état de l'art des techniques de codage par descriptions multiples. Les techniques présentées sont classées suivant leur position dans le schéma de codage. Nous considérons un schéma de codage simple et classique, composé de deux parties, le codage de source et le codage de canal. La partie codage de source est divisée en trois étapes : une transformation, une quantification et un codage entropique. Cette chaîne de codage est représentée sur la figure 1.3.

FIG. 1.3 – *Exemple de schéma de compression classique.*

Le codage par descriptions multiples est par nature une technique de codage conjoint source/canal. La majorité des techniques présentées interviennent donc au niveau du codage de source. Nous verrons cependant qu'il existe une technique qui intervient au niveau du codage de canal et qui construit les descriptions indépendamment du codage de source, avec tout de même une contrainte de progressivité sur ce dernier.

1.3.1 Codage par descriptions multiples basé transformation

Dans la chaîne de codage de la figure 1.3, la transformation a pour but d'une part de décorréler les données, et d'autre part d'obtenir une représentation de ces données compacte et qui supporte la suppression d'une partie de l'information au prix d'une perte de qualité acceptable lors de la reconstruction. Dans le cas du codage par descriptions multiples, nous distinguons deux approches. La première consiste à remplacer la transformation classique par une autre transformation qui permet à la fois de générer les descriptions et de contrôler la redondance entre ces descriptions. C'est le cas par exemple de la transformée multi-ondelettes (*multi-wavelet transform, MWT*) ou des transformées orthogonales recouvrantes (*lapped orthogonal transform, LOT*). La seconde approche consiste à insérer entre le transformation classique et la quantification une autre transformation qui aura pour but de rajouter de la corrélation et de permettre la création des descriptions. On peut citer comme exemples la transformation par appariement de doublets (*pairwise correlating transform, PCT*) ou les expansions sur bases de fonctions redondantes (*frame expansions*).

1.3.1.1 Transformée orthogonale recouvrante

Les *lapped orthogonal transform (LOT)* sont des transformations par blocs recouvrantes. Dans un schéma de codage par blocs de type JPEG, la transformation DCT peut être remplacée par une LOT sur des blocs qui se chevauchent partiellement. Ce principe a été utilisé initialement pour éviter les effets de blocs [MS89] [TY01], qui sont un artefact visuel propre à ce type de schémas de compression. Par la suite, Hemami à proposé de concevoir des LOTs adaptées au codage robuste d'image [Hem96]. Dans un schéma de codage par blocs, lorsqu'un bloc est perdu, il peut être estimé à partir des blocs voisins en utilisant des techniques de traitement du signal et en s'appuyant sur des propriétés connues des images. La propriété de recouvrement des LOTs permet en plus d'introduire dans chaque bloc un peu d'information sur ses blocs voisins. Cette information pourra alors être exploitée lors de l'estimation.

Dans [Hem96], Hemami résout le problème de la conception de LOTs qui ont la propriété à la fois de minimiser le coût de codage et de maximiser les performances de la reconstruction pour un algorithme d'estimation donné. Des familles de LOTs peuvent alors être créées en fonction des caractéristiques de pertes du canal, de l'efficacité de reconstruction désirée et de la compression désirée. Ces travaux ont été prolongés dans [CW00]. Un schéma de codage par descriptions multiples qui utilise les LOTs peut être créé en répartissant les blocs codés alternativement dans les différentes descriptions, de manière à ce que la perte d'une description n'entraîne pas la perte de deux blocs voisins spatialement [CW98].

Les techniques de codage par LOTs sont peu utilisées actuellement. En effet, les transformations par blocs de type DCT ou LOT sont délaissées au profit de la transformée en ondelettes.

1.3.1.2 Transformée multi-ondelettes

Les multi-ondelettes sont une généralisation récente des ondelettes. Par nature, elles sont adaptées au traitement de signaux multi-canaux. La conception de bancs de filtres multi-ondelettes basée sur les "multi-ondelettes équilibrées" est présentée dans [LV98a] et [LV98b]. Pour résumer, les bancs de filtres multi-ondelettes sont des bancs de filtres à entrées multiples et à sorties multiples, comme cela est présenté par exemple sur la figure 1.4 pour deux canaux. Pour pouvoir être traité par un tel banc de filtres, un signal mono-dimensionnel doit d'abord être vectorisé. Une technique simple consiste à décomposer le signal en composantes polyphases. cela conduit au banc de filtres de la figure 1.5. Pour un traitement détaillé de ce sujet, se reporter à [LV98a].

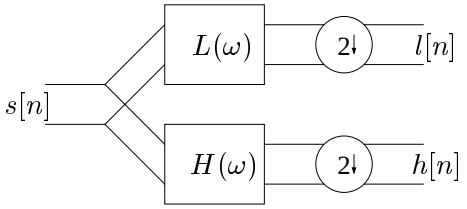


FIG. 1.4 – *Banc de filtres multi-ondelettes.*

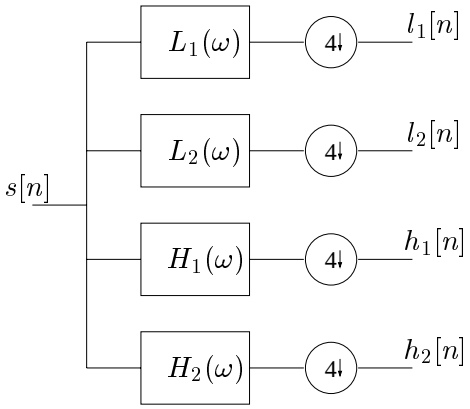


FIG. 1.5 – *Reformulation du banc de filtres multi-ondelettes pour un signal mono-dimensionnel.*

Ce travail constitue une base adaptée à une application au codage par descriptions multiples. Dans [CGA01], le principe des bancs de filtres multi-ondelettes est appliqué au codage d'images par descriptions multiples pour une transmission progressive. Un banc de filtres du type de celui présenté sur la figure 1.5 est utilisé pour créer les deux descriptions. Les sous-bandes passe-bas de ces dernières peuvent être décomposées par une transformée en ondelettes classique. Chaque description est ensuite codée avec l'algorithme EBCOT [Tau00], qui a la propriété de créer des trains binaires progressifs.

Lorsqu'une description est perdue, des techniques basées sur le filtrage de Wiener sont utilisées pour estimer la description manquante. Les performances de ce schéma sont ensuite améliorées en incluant dans chaque description une information complémentaire sur l'autre description.

1.3.1.3 Codage en sous-bandes

Nous avons présenté deux techniques de codage par descriptions multiples basées sur des transformations dont le but est de contrôler la décorrélation des données. La redondance conservée par ces transformations entraîne une perte en compression, mais permet une meilleure reconstruction des données en cas de perte d'une description. Toute la difficulté consiste alors à concevoir des transformations qui optimisent le compromis entre gain de codage et efficacité de reconstruction, en fonction des caractéristiques de pertes du canal. Nous nous intéressons à présent aux *transformations redondantes*, dont le seul but est de rajouter de la redondance dans des données décorrélées. Dans le schéma de la figure 1.3, elles s'insèrent entre la transformation et la quantification. Ces transformations sont conçues de manière à minimiser une erreur globale de reconstruction, en présence d'erreurs de transmission ou de pertes. Les critères de gain de codage ne sont plus considérés.

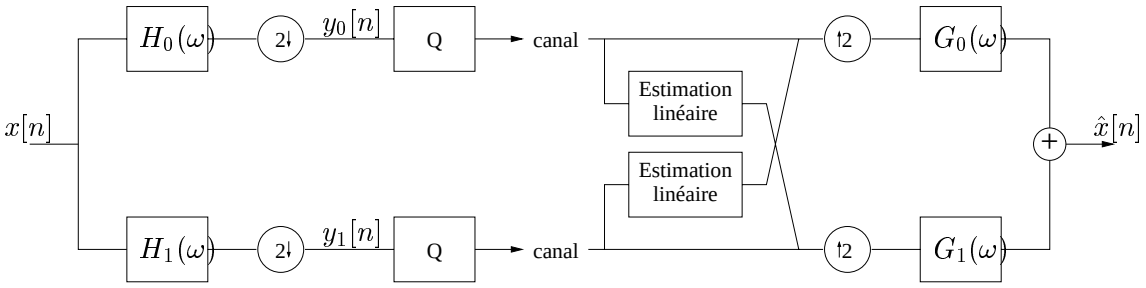


FIG. 1.6 – *Banc de filtres à descriptions multiples.*

Le première technique que nous présentons consiste à synthétiser des bancs de filtres à descriptions multiples. Le problème de la décomposition optimale en sous-bandes pour le codage par description multiples a été résolu simultanément par Yang et Ramchandran [YR98] [YR00] et Dragotti, Servetto et Vetterli [DSV00] [DSV02], dans le cas de bancs de filtres à deux canaux, comme cela est illustré sur la figure 1.6. L'approche proposée dans [YR98] et [YR00] consiste à minimiser la distorsion obtenue avec une seule description pour un niveau de redondance donné. L'optimisation repose sur une formulation Lagrangienne qui est résolue directement dans le domaine fréquentiel pour produire les réponses des filtres. Lorsqu'une description est perdue, elle est estimée à partir de la description reçue par un filtre de Wiener. Les auteurs remarquent que pour une redondance nulle, leur solution rejoint le cas d'une optimisation de gain de codage classique. En revanche, pour une redondance maximale, leur optimisation aboutit à une transformée polyphase triviale.

Cette technique est peu utilisée dans des systèmes de codage pratiques. Sa capacité d'adaptation à des conditions de transmission variables est limitée par la nécessité de synthétiser un banc de filtres pour chaque niveau de redondance.

1.3.1.4 Transformations par appariement de doublets

Dans [OWVR97] les auteurs décrivent deux méthodes qui permettent de créer des transformations à descriptions multiples pour le codage de sources gaussiennes. Ces transformations sont appliquées aux sorties de transformations décorrélantes comme la KLT (Karhunen-Loève Transform) [OWVR97], ou la DCT [RPJ+99]. Les variables d'entrée des transformations sont considérées comme gaussiennes, indépendantes et de variances différentes. Ici une transformation $2 * 2$ est appliquée à une paire de coefficients d'entrée et génère en sortie deux coefficients corrélés qui seront envoyés sur deux canaux différents. Deux types d'appariements sont considérés: l'appariement orthogonal et l'appariement non-orthogonal. Les auteurs introduisent une métrique permettant d'évaluer les performances des schémas de codage par descriptions multiples : la courbe de redondance débit-distorsion. Cette courbe permet de quantifier la redondance et de mesurer l'efficacité du schéma à réduire la distorsion lors de la réception d'une seule description. La distorsion engendrée par la réception des deux canaux est considérée comme fixée.

Considérons deux variables aléatoires indépendantes gaussiennes A et B de variances σ_a^2 et σ_b^2 respectivement. Soit T une transformation d'appariement orthogonale générant deux variables aléatoires C et D.

$$\begin{bmatrix} C \\ D \end{bmatrix} = T \begin{bmatrix} A \\ B \end{bmatrix} \tag{1.10}$$

La transformation T contrôle la redondance en faisant varier la corrélation entre C et D. Soit T paramétrée par θ:

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{1.11}$$

La corrélation est paramétrée par l'angle ϕ, où $E\{CD\} = \sigma_C \sigma_D \cos\phi$. On peut alors montrer que

$$\cot\phi = -\frac{1}{2}\sin(2\theta)\frac{(1-r^2)}{r} \tag{1.12}$$

où $r = \frac{\sigma_a}{\sigma_b}$.
Si un seul canal est reçu, D (ou C) peuvent être estimés grâce au $\tilde{C}$ reçu (respectivement $\tilde{D}$) à l'aide des estimateurs linéaires optimaux:

$$\hat{D}(\tilde{C}) = \frac{\sigma_d \sigma_c}{\sigma_c^2 + \sigma_q^2}\cos(\phi)\tilde{C} \quad \text{et} \quad \hat{C}(\tilde{D}) = \frac{\sigma_c \sigma_d}{\sigma_d^2 + \sigma_q^2}\cos(\phi)\tilde{D} \tag{1.13}$$

Ici, σ_q^2 est l'énergie de l'erreur de quantification générée par les deux canaux. La distorsion pour un seul canal D_1 reçu (définie comme l'erreur de reconstruction moyenne

par variable) peut alors s'exprimer en fonction de ϕ et des variances σ_c^2, σ_d^2 et σ_q^2. Soit ρ la redondance accordée par le schéma MDC par rapport à un schéma de codage à simple description :

$$\rho = -log_2 \sin(\phi) \tag{1.14}$$

On peut ainsi définir la courbe redondance débit-distorsion du codeur MDC.

$$D_{1,ORTH}(\rho) = \frac{K_q(\rho)}{2} \frac{\sigma_a^2 + \sigma_b^2}{2} 2^{-2\rho} \tag{1.15}$$

où $K_q(\rho) = 1 + \sigma_q^2[\frac{2}{\sigma_a^2+\sigma_b^2} + (\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2})(1 - 2^{-2\rho})]$. On devra déterminer la valeur minimale de cette courbe pour trouver la transformation optimale. On peut remarquer que si σ_q^2 est très inférieure à σ_b^2 et σ_a^2 alors $K_q(\rho)$ sera peu différent de 1.
Le cas des transformations non-orthogonales du type:

$$T = \begin{bmatrix} 1 & \beta \\ -\frac{1}{2\beta} & \frac{1}{2} \end{bmatrix} \tag{1.16}$$

a aussi été traité. Dans ce cas la courbe redondance débit-distorsion est donnée par:

$$D_{1,NON-ORTH} = \frac{1+\beta^2}{8\beta^2}(\sigma_a^2 + \beta^2\sigma_b^2)2^{-2\rho} \tag{1.17}$$

Les auteurs proposent dans un deuxième temps une méthode permettant de trouver l'appariement de N_{coeff} coefficients (N_{coeff} pair) de manière à trouver la combinaison de $N_{coeff}/2$ paires minimisant la distorsion D_1.

Dans [WOR98] les auteurs étudient les courbes redondance débit-distorsion de transformations arbitraires. Ils montrent que l'unique transformation optimale permettant d'atteindre un niveau de redondance donné est obtenu par changement d'échelle de la transformation non-orthogonale proposée dans [OWVR97]. Une nouvelle étude de l'appariement optimal de N_{coeff} coefficients en $N_{coeff}/2$ paires est menée. Le résultat est que la méthode d'appariement donnant les meilleurs résultats sur l'ensemble des valeurs de redondance possibles est celle qui apparie la k^{eme} plus grande variable avec la $(N_{coeff} - k)^{eme}$ plus grande variable.

Dans [GK98], Goyal et al. proposent une généralisation des travaux de [OWVR97]. L'idée est d'augmenter le nombre de descriptions générées. Soit n_d le nombre de descriptions générées et soit l le nombre de canaux perdus ($l \leq n_d$). D_l est la distorsion obtenue lorsque l descriptions sont perdues. Cette dernière est calculée par une moyenne pondérée effectuée sur les $C_{n_d}^l$ combinaisons de l pertes possibles. La distorsion globale finale est donnée par la somme pondérée des distorsions obtenues en faisant varier le nombre de canaux disponibles:

$$\bar{D} = \sum_{l=0}^{n_d} \alpha_l D_l \tag{1.18}$$

α_l est une pondération qui peut prendre en compte les probabilités de perdre un certain nombre de descriptions. Le but des auteurs est alors, étant donné une contrainte de débit R, de trouver la transformation permettant de minimiser $\bar{D}$. Cette étude appliquée au cas des transformations $2 * 2$ permet d'aboutir à la transformation optimale dans le cas général de probabilités de pertes différentes sur chaque canal. Il est ensuite aisé de déduire la transformation optimale dans le cas de probabilités de pertes égales. Les auteurs montrent aussi que la transformation optimale proposée par Orchard et al. dans [OWVR97] n'est en fait qu'une des transformations optimales parmi un ensemble de transformations plus vaste. D'autre part, les auteurs montrent comment obtenir deux descriptions de débits égaux. Comme le montrent les auteurs, l'augmentation du nombre de descriptions complexifie le problème. Ils proposent toutefois une méthode applicable à un nombre pair de descriptions. Celle-ci consiste à cascader les transformations telle que dans la figure 1.7.

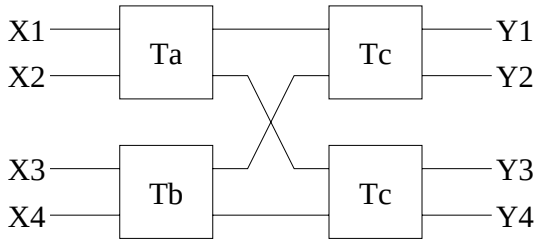


FIG. 1.7 – *Structure en cascade permettant de multiplier le nombre de descriptions obtenues par PCT.*

1.3.1.5 Transformations sur bases de fonctions redondantes

Les transformations sur bases de fonctions redondantes constituent une base théorique adaptée à une application au codage par descriptions multiples. Dans certains cas, ces transformations peuvent s'apparenter à des méthodes de codage de canal. Une des limitations des codes correcteurs classiques (FEC) est qu'ils risquent de conduire à une sous utilisation de la bande passante disponible. Supposons en effet une adaptation de la source aux conditions du canal amenant celle-ci à envoyer $\mathcal{N}$ paquets, parmi lesquels $\mathcal{K}$ paquets contiennent des données source et $\mathcal{N} - \mathcal{K}$ de la redondance générée par un code de Reed-Solomon systématique. L'utilisation du canal sera optimale si exactement $\mathcal{K}$ paquets sont reçus. Par contre si on reçoit plus de $\mathcal{K}$ paquets, les paquets supplémentaires n'ajoutent aucune information et inversement si moins de $\mathcal{K}$ paquets sont reçus, on ne pourra utiliser que les paquets sources.

Une alternative à ce type de schéma a été proposée dans [GKV98][GK98]. Cette approche s'appuie sur la théorie des expansions sur bases de fonctions redondantes. Goyal et al. en s'appuyant sur une généralisation du codage par descriptions multiples, proposent de décrire une source par $\mathcal{N}$ paquets de sorte que n'importe quel sous ensemble de paquets est utilisable. Ils utilisent pour cela un opérateur d'expansion F_{exp}

de $\mathbb{C}^{\mathcal{K}}$ dans $\mathbb{C}^{\mathcal{N}}$ associé à une base de fonctions redondantes dite "tight" normalisée $\Phi = \{\varphi_m\}_{m=1}^{\mathcal{N}} \subset \mathbb{C}^{\mathcal{K}}$ telle que $\|\varphi_m\| = 1$. Les $\mathcal{N}$ composantes de $\hat{y} = Q(F_{exp}(x))$ sont transmises comme étant différentes descriptions de x. $Q(.)$ représente une quantification scalaire suivie d'un codage entropique. Les auteurs analysent la distorsion obtenue lorsque e descriptions sont perdues.

Soit $\mathcal{E}$ l'ensemble des index des effacements, autrement dit supposons que les projections $\{\langle x, \varphi_m \rangle\}_{m \in \mathcal{E}}$ sont effacées. Si $\Phi^{'} = \Phi \backslash \{\varphi_m\}_{m \in \mathcal{E}}$ est une base de fonctions redondantes, l'estimation $\hat{x}$ de x qui minimise l'EQM est obtenue avec la base duale de $\Phi^{'}$; sinon x ne peut être estimé qu'à partir d'un sous-espace et une connaissance de sa distribution est nécessaire pour obtenir une bonne estimation. La quantification est modélisée par un bruit blanc additif η, tel que $\eta = \hat{y} - F_{exp}x$ ait des composantes indépendantes et soit indépendant de x avec $E[\eta_m]^2 = \sigma^2$.

Quand il n'y a aucun effacement, l'erreur entre le signal reconstruit $\hat{x}$ et le signal original x est uniquement due aux erreurs de quantification. Dans ce cas l'EQM est donnée par:

$$EQM_0 = \frac{\mathcal{K}\sigma^2}{\mathcal{N}} \tag{1.19}$$

On constate alors qu'en théorie, une réduction de l'erreur de quantification est obtenue.

Une application à l'image est proposée dans [GKAV98]. Une alternative à un code en bloc générant 2 paquets de redondance à partir de 8 paquets de données de source est recherchée. On le remplace par une transformation $10 * 8$ correspondant à une DFT réelle de longueur 10 appliquée à une séquence de taille 8. Cette transformation peut être construite sous la forme $F_{exp} = [F_{exp}^{(1)} F_{exp}^{(2)}]$ avec:

$$\begin{array}{rcl} F_{exp_{ij}}^{(1)} & = & \frac{1}{2}\cos\frac{\pi(i-1)(2j-1)}{10} \\ F_{exp_{ij}}^{(2)} & = & \frac{1}{2}\sin\frac{\pi(i-1)(2j-1)}{10} \quad \text{avec} \quad 1 \leq i \leq 10,\ 1 \leq j \leq 4 \end{array} \tag{1.20}$$

Cette transformation est appliquée à des coefficients DCT. Le codage se déroule de la manière suivante.

1. Une DCT est appliquée aux blocs 8*8 de l'image
2. Des vecteurs de taille 8 sont formés à partir des coefficients DCT de fréquences similaires séparées spatialement.
3. On applique la transformation F_{exp} à chaque vecteur.
4. Chaque vecteur de longueur 10 est quantifié uniformément avec un pas dépendant de la fréquence.

La méthode est comparée à une approche utilisant des FEC sous la forme d'un code en bloc systématique générant des vecteurs de 10 coefficents (8 coefficients de source et 2 coefficients de redondance) à partir des vecteurs de 8 coefficients DCT. Dans les deux approches, les éléments des vecteurs de taille 10 générés sont répartis dans 10 paquets différents. Les tests montrent que la méthode basée transformation linéaire l'emporte dès que plus de 2 descriptions sont perdues. Par contre, contrairement à l'attente des auteurs, aucune amélioration n'est constatée lors de la réception de 10 et 9 paquets.

L'application des expansions sur bases de fonctions redondantes au codage par descriptions multiples a été étudiée plus en détail par Hénocq [Hén02] dans ses travaux de thèse. Il montre que cette approche permet non seulement la correction des pertes mais aussi la réduction des erreurs de quantification. Il tire alors effectivement partie de tous les paquets reçus et non seulement de $\mathcal{K}$ paquets au maximum.

1.3.1.6 Techniques spectrales

Cette section aborde une autre forme de codage redondant basée sur des interpolations spectrales. Ces techniques popularisées par Blahut [Bla84] découlent directement des méthodes de codage canal. Elles sont toutefois transposées dans le domaine des réels. Ces techniques peuvent être vues comme un cas particulier de transformation sur bases de fonctions redondantes [Hén02].

Blahut montre dans les années 80 que les codes cycliques peuvent être définis comme des mots de codes ayant certaines composantes spectrales spécifiées égales à 0 [Bla84]. Suivant ce principe un code BCH de pouvoir de correction t est l'ensemble de tous les vecteurs ayant un ensemble fixé de $2t$ composantes consécutives égales à 0 [Bla92]. Une manière de coder un code BCH est donc de le faire dans le domaine fréquentiel, comme illustré sur la figure 1.8, dont les éléments s'insèrent entre la transformation et la quantification dans le schéma de codage de la figure 1.3. Ce type de codes est

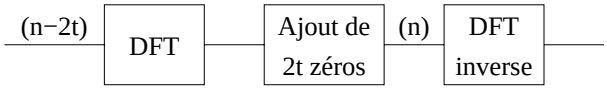


FIG. 1.8 – *Redondance introduite par interpolation spectrale.*

aussi appelé codes DFT (Discret Fourier Transform), du fait de l'utilisation dans le processus de codage d'une DFT et d'une DFT inverse. Notons que Wu et Shiu propose dans [WS95] une variante des codes DFT basée cette fois sur l'utilisation de la DCT.

Ce type de codes correcteurs possède de nombreux avantages par rapport aux codes classiques. Ils n'ont pas de contraintes sur la taille des codes et permettent des implémentations faciles et efficaces. Ils sont mis en oeuvre dans un certain nombre d'articles tels que dans [GDR00] pour la robustification d'une transmission sans fils et dans [MHET99] pour la robustification d'une transmission audio sur ATM. Une variante des codes DFT est aussi proposée dans [Roz99]. Toutefois ici les zéros sont insérés dans le domaine temporel, ce qui permet d'obtenir des racines dans le domaine fréquentiel.

Les avantages des codes DFT sont toutefois à nuancer, car contrairement aux codes correcteurs classiques, ces codes sont sensibles au bruit de quantification et au bruit d'arrondi.

Marvasti et al. proposent dans [MHET99] une étude des effets du bruit de quantification sur un système de robustification contre les effacements basé sur les codes DFT. Les auteurs montrent que la sensibilité au bruit de fond dépend de la position des pertes. En effet alors que les effets du bruit de fond sont quasi nuls en présence de pertes isolées, ceux-ci s'accroissent fortement dès que les pertes sont situées à des

positions consécutives. Lorsque les pertes sont consécutives, les valeurs des coefficients du polynôme localisateur d'erreurs seront très élevées. Cela signifie que la moindre erreur sur les syndromes (erreur introduite par le bruit de quantification par exemple) est fortement amplifiée. Dans ces conditions les estimations des valeurs des racines du polynôme localisateur d'erreur s'écartent fortement des racines réelles. On obtient alors de fortes erreurs lors de l'estimation de l'amplitude des erreurs.

On trouve dans la littérature deux types de solution au problème posé par le bruit de fond. La première, proposée notamment dans [MHET99], consiste à limiter l'effet du bruit de fond en tentant d'empêcher les configurations de pertes problématiques de se produire. Les auteurs observent que la nature des configurations problématiques (les pertes consécutives pour les codes DFT) est liée au noyau de transformation utilisé (ici, le noyau DFT). Par conséquent, on pourra supprimer ces configurations problématiques en changeant de noyau de transformation. Toutefois, le changement de noyau n'a pas pour effet de réduire le nombre de configurations problématiques. Ces configurations, connues dans le cas du noyau DFT, seront en effet remplacées par d'autres configurations plus aléatoires. Néanmoins, ce type d'approche reste intéressant car on peut imaginer que le choix du noyau soit dépendant du modèle régissant les pertes sur le canal. Il est clair, par exemple, que le noyau DFT n'est a priori pas adapté à des pertes en rafales.

Le second type de méthode ne tente pas d'empêcher les configurations problématiques, mais plutôt de distinguer lors du décodage les erreurs de transmission des erreurs dues au bruit de fond. Comme dans [WS95] et [Gab01], l'algorithme de décodage peut être modifié pour prendre en compte la présence de bruit de fond. Il permet en effet d'introduire facilement une connaissance de la contribution statistique du bruit de quantification et ainsi de tenter de faire la distinction entre les erreurs dues à des pertes et le bruit de fond.

1.3.1.7 Transformée polyphase et quantification sélective

L'utilisation de la transformée polyphase pour créer deux descriptions peut être vue comme la simplification d'autres méthodes déjà présentées, telles que la transformée multi-ondelettes (section 1.3.1.2) et le codage en sous-bandes (section 1.3.1.3), dans le cas de la redondance maximale, ou bien l'appariement de doublets (section 1.3.1.4), dans le cas où aucune corrélation n'est ajoutée avant de répartir les paires de coefficients entre les deux descriptions. Dans [JO99], Jiang et Ortega étudient spécifiquement le cas de la transformée polyphase, en la couplant au mécanisme de quantification sélective pour contrôler la redondance. Ils s'inspirent d'un premier travail appliqué à la transmission de signaux audio sur des réseaux sujets aux pertes de paquets [HSHW95]. Le système proposé à l'avantage de simplifier le codage et le décodage et de réduire la complexité par rapport aux autres techniques de codage par descriptions multiples.

Soit donc le système représenté par la figure (1.9). L'entrée X est décomposée en deux sous-sources Y_1 et Y_2 par une transformation polyphase. Chacune des composantes est quantifiée séparément par Q_1 pour constituer l'information primaire du canal correspondant. Pour reconstruire le deuxième canal lorsque celui-ci est perdu,

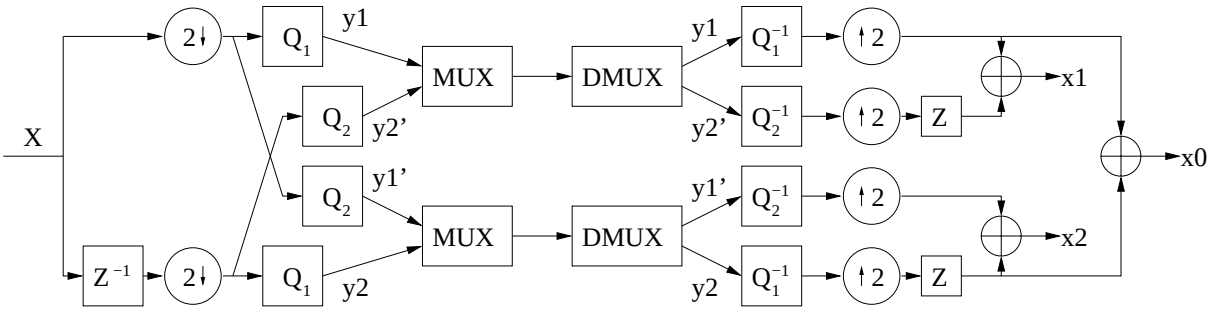


FIG. 1.9 – *Codage par transformée polyphase et quantification sélective.*

chaque canal transporte aussi une version quantifiée grossièrement de l'autre canal. Au récepteur, si les deux canaux sont reçus, on obtient des données finement quantifiées pour les deux composantes polyphases. Si un canal est perdu, une composante polyphase est reçue finement quantifiée, l'autre étant quantifiée grossièrement. On a donc une séparation explicite de la composante polyphase primaire et de la composante polyphase grossièrement quantifiée représentant la redondance pour le deuxième canal, ce qui simplifie la mise en oeuvre.

Une optimisation du rapport entre le débit de l'information primaire et celui de l'information de redondance sur un même canal est proposée. Cette optimisation permet de minimiser à la fois la distorsion centrale D_c et les distorsions latérales $D_{l,i}$.

Supposons chaque description codée au même débit R_0. Soit X une source aléatoire i.i.d. de moyenne nulle, de densité de probabilité $p(x)$ et de variance σ^2. Avec l'hypothèse de quantification haute résolution, on peut approximer $D_c = h\sigma^2 2^{-2R_O}$ avec h une intégrale constante telle que $h = \frac{1}{12}\{\int_{-\infty}^{\infty}[p(x)]^{\frac{1}{3}}\,dx\}^3$. Pour une source gaussienne, $h = \frac{\sqrt{3}\pi}{2}$.

Si R_0 est le débit de l'information primaire et ρ celui de l'information de redondance, on peut exprimer les distorsions latérales et le débit effectif par:

$$\begin{array}{rcl} D_{l,i} & = & \frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2R_O} \\ R_{l,i} & = & \frac{1}{2}(R_0 + \rho) \\ i & = & 0, 1 \end{array} \tag{1.21}$$

la distorsion centrale correspondante est:

$$D_c = h\sigma^2 2^{-2R_O}. \tag{1.22}$$

Le débit total est $R = R_0 + \rho$.

Le problème de partage de débit entre l'information primaire et la redondance est résolu par une optimisation sous contrainte. En utilisant les multiplieurs de Lagrange, on définit une fonction de coût J:

$$\begin{array}{rcl} J & = & D_c + \lambda D_l \\ & = & h\sigma^2 2^{-2(R-\rho)} + \lambda(\frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2(R-\rho)}) \end{array} \tag{1.23}$$

On trouve le débit de redondance optimal en dérivant J par rapport à ρ.

$$\rho^* = \frac{1}{2}R + \frac{1}{4}\log_2(\frac{\lambda}{2+\lambda}) \tag{1.24}$$

On remarque donc que ρ se situe entre 0 et $\frac{1}{2}R$. Avec ρ proche de 0 on privilégie la distorsion centrale alors qu'avec ρ proche de $\frac{1}{2}R$ les distorsions latérales sont privilégiées.

Les auteurs montrent que leur schéma permet d'obtenir des résultats comparables à ceux du schéma MDSQ de [Vai93]. Ils démontrent aussi qu'une meilleure quantification permettra d'accroître la qualité des résultats en les approchant des limites d'Ozarow (section 1.2.1).

Dans une deuxième partie les auteurs introduisent des connaissances du canal dans les mesures de distorsion. Les canaux sont modélisés par des canaux à effacement indépendants et sans mémoire de probabilité de perte P_e. La distorsion moyenne au récepteur est alors:

$$\begin{aligned} D &= (1-P_e)^2 D_c + 2P_e(1-P_e)D_s + P_e^2 D_p \\ &= (1-P_e)^2 h\sigma^2 2^{-2R_0} + P_e(1-P_e)[h\sigma^2 2^{-2\rho} + h\sigma^2 2^{-2R_0}] + P_e^2\sigma^2 \end{aligned} \tag{1.25}$$

L'allocation optimale de débit est donnée par:

$$\rho^* = \frac{1}{2}R + \frac{1}{4}\log_2(P_e) \tag{1.26}$$

On remarque que pour des canaux subissant de fortes pertes ($P_e \sim 1$) la meilleure approche est de partager équitablement le débit entre le codage primaire et la redondance. Par contre, pour un canal parfait, il est inutile d'utiliser de la redondance. Ce travail a été approfondi dans [PA00].

Les auteurs ont ensuite testé l'apport d'une nouvelle politique de quantification. Le codeur de Said-Pearlman [SP96] est utilisé pour la quantification et le codage entropique des composantes polyphases. Une comparaison avec le schéma basé MDSQ de Servetto et al. [SRVN00] montre un avantage certain pour le schéma de Jiang et Ortega. Ce test permet aussi de comparer l'apport d'une organisation vectorielle des données comparativement à l'organisation scalaire du départ. Les vecteurs sont formés des coefficients ondelettes correspondant à la même position spatiale mais provenant de sous-bandes différentes. C'est en fait une extension de l'algorithme SPIHT de Said et Pearlman séparant le Zerotree en composantes polyphases. Les résultats escomptés ne sont apparemment pas au rendez-vous puisque la source scalaire donne les meilleurs résultats.

La simplicité et l'efficacité du principe du codage par descriptions multiples basé sur la transformée polyphase, avec ou sans quantification sélective, a conduit au développement de nombreuses applications. Dans [JO02] et [LSG01], la transformée polyphase est appliquée au codage de parole par descriptions multiples. Dans [GSK01], la transformée polyphase est appliquée directement dans le domaine spatial pour une application au codage vidéo. Dans [Apo00] et [AW01], le codage vidéo est également considéré, mais la transformée polyphase est appliquée temporellement. Le schéma ainsi obtenu est

remarquablement simple et efficace. Le codage d'image est considéré dans [LT01], où la transformée polyphase est combinée avec un pré-traitement spécifique au codage et une technique d'interpolation orientée contours au décodage. Dans [FFL02], le codage d'image est également considéré et l'utilisation de la transformée polyphase est mise en avant. Cependant, l'ajout de redondance par addition de zéros dans le domaine DCT rapproche plutôt cette étude des techniques spectrales (section 1.3.1.6). Enfin, la technique proposée dans [CAK01] peut être assimilée au principe de la transformée polyphase, cette dernière étant alors appliquée par blocs.

Bien qu'étant une technique de codage par descriptions multiples basée transformation, la transformée polyphase, lorsqu'elle est associée à la quantification sélective, se rapproche des techniques basées quantification. Nous considérons celles-ci dans la section suivante.

1.3.2 Codage MD basé quantification

En 1993, Vaishampayan [Vai93] a introduit la quantification scalaire à descriptions multiples (*multiple description scalar quantization, MDSQ*). Cet article de référence a servi de base à de nombreux développements [VD94], dans le but d'étudier les performances théoriques de cette technique [BV97] [VB98] [VBC98] ou d'en améliorer les performances pratiques [BWR99]. Plusieurs applications ont été proposées [YV95] [SRVN00]. Par la suite, le principe du codage par descriptions multiples basé quantification à été étendu à la quantification codée en treillis (*Trellis coded quantization, TCQ*) [JT99] et à la quantification vectorielle (*multiple description vector quantization, MDVQ*) [VSS01] [DSV02].

1.3.2.1 Quantification scalaire par descriptions multiples

La quantification scalaire par descriptions multiples (MDSQ) est utilisée essentiellement pour le codage à deux descriptions. Une première approche de la MDSQ consiste à utiliser deux quantificateurs uniformes dont les régions de décision sont décalées l'une par rapport à l'autre d'un demi intervalle, comme sur la figure 1.10-(a).

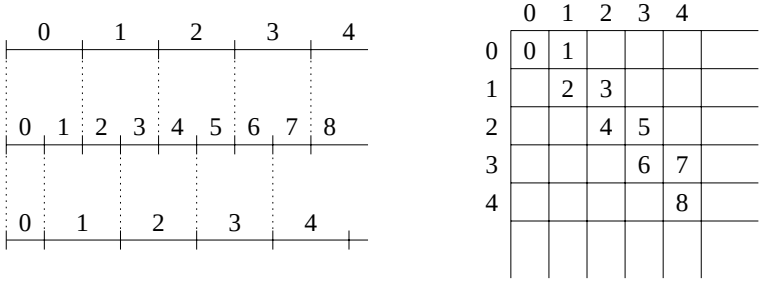


FIG. 1.10 – *(a) Quantification décalée, (b) représentation matricielle : les indices des quantificateurs grossiers correspondent aux colonnes et aux lignes.*

Ainsi, si chaque quantificateur a un débit de R bits par échantillons de signal, l'erreur de reconstruction est équivalente à l'erreur qui aurait été obtenue avec un quantificateur unique avec un débit de $R+1$ bits, lorsque les deux descriptions sont reçues. Si seule l'une des deux descriptions est reçue, alors l'erreur de reconstruction est équivalente à l'erreur obtenue avec un quantificateur induisant un débit de R bits. Donc, avec un tel quantificateur MDSQ, en l'absence de pertes, il est nécessaire de transmettre $2R$ bits pour obtenir des performances équivalentes à un quantificateur avec un débit de $R+1$ bits. Une telle technique est peu efficace et peu flexible. En s'appuyant sur la représentation matricielle de la figure 1.10-(b), on peut définir la MDSQ de façon plus générale.

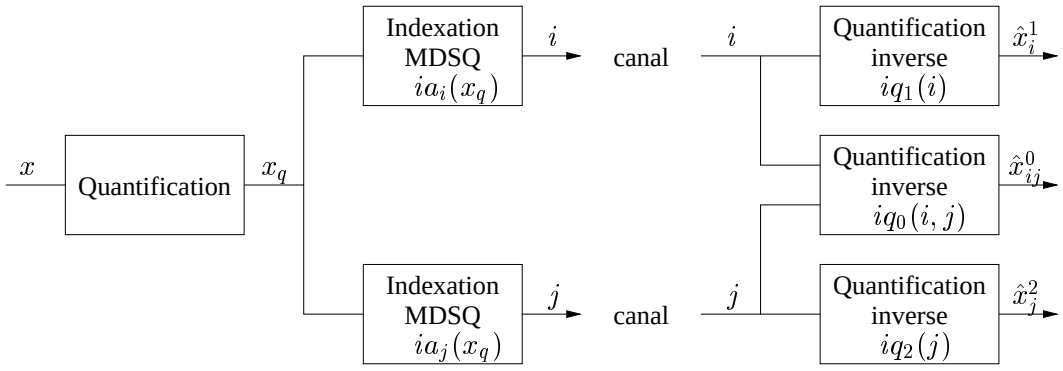


FIG. 1.11 – *Quantification scalaire par descriptions multiples*

Considérons le cas d'une transmission sur deux canaux, comme illustrée sur la figure (1.11). La quantification scalaire par descriptions multiples à (M_1, M_2) niveaux de reconstruction va projeter les échantillons du signal x sur les niveaux de reconstruction $\hat{x}_{ij}^0$, $\hat{x}_i^1$, et $\hat{x}_j^2$, prenant respectivement leurs valeurs dans les ensembles $\hat{X}^0 = \left\{\hat{x}_{ij}^0, (i,j) \in \mathcal{C}\right\}$, $\hat{X}^1 = \left\{\hat{x}_i^1, i \in I_1\right\}$ et $\hat{X}^2 = \left\{\hat{x}_j^2, j \in I_2\right\}$. Les ensembles I_1, I_2 sont tels que $I_1 = \{1, 2, \ldots, M_1\}$, $I_2 = \{1, 2, \ldots, M_2\}$. L'ensemble $\mathcal{C}$ est un sous-ensemble de $I_1 \times I_2$. Le quantificateur MDSQ est alors constitué de deux quantificateurs, $Q_1 : \mathbb{R} \to I_1$ et $Q_2 : \mathbb{R} \to I_2$, qui sélectionnent les index i et j. Au niveau du décodeur il comprend trois quantificateurs inverses $IQ_0 : \mathcal{C} \to \mathbb{R}$, $IQ_1 : I_1 \to \mathbb{R}$ et $IQ_2 : I_2 \to \mathbb{R}$, dont les sorties sont les niveaux de reconstruction prenant leurs valeurs dans les dictionnaires $\hat{X}^0$, $\hat{X}^1$, $\hat{X}^2$ et correspondant respectivement aux index ij, i et j. Le débit du codeur Q_m est donné par $R_m = log_2 M_m$ bits par échantillons du signal de source, avec $m = 1, 2$.

Les deux codeurs imposent une partition $A = \{A_{ij}, (i,j) \in \mathcal{C}\}$ de l'ensemble $\mathbb{R}$, telle que $A_{ij} = \{x / Q_1(x) = i, Q_2(x) = j\}$. Cette partition est définie arbitrairement par une *table d'index*, telle que celle de la figure 1.10-(b). L'allocation des index doit être telle que si les deux index i et j sont reçus, la qualité de reconstruction du signal est équivalente à une quantification fine, et si seul l'un des deux index est reçu, le niveau de reconstruction est équivalent à une quantification à pas "grossier". Un exemple de table d'index est représenté sur la figure 1.12. La redondance entre les deux descriptions

j \ i	1	2	3	4	5	6	7	8
1	1	3	6					
2	2	5	7	9				
3	4	8	10	12	14			
4		11	13	15	17	20		
5			16		19	22	25	
6				18	21	24	27	28
7					23	26	29	31
8						30	32	33

FIG. 1.12 – *Table d'index MDSQ linéaire modifiée [Vai93].*

dépend du nombre de cellules occupées dans la table d'index. Plus elles sont nombreuses, moins la redondance est importante. Soit N le nombre de cellules occupées dans la table d'index. Le rendement équivalent à un code de canal d'une telle table d'index est $log_2(N)/log_2(M_1 \times M_2)$. L'index de la valeur de reconstruction centrale associée à la réception des deux index $i = 5$ et $j = 6$ est 21. Si seul l'index $i = 5$ est reçu, alors on sait que la valeur quantifiée appartient à l'union des intervalles étiquetés 14, 17, 19, 21 et 23. On remarque que l'union de ces intervalles forme un intervalle disjoint. La conception de la table d'index est un élément clé de la MDSQ. Elle en conditionne les performances. Une recherche exhaustive de la meilleure combinaison est impossible, car trop complexe. Il est donc nécessaire de définir des critères qui permettent de concevoir les tables d'index. La dispersion d'une table d'index est définie par la valeur maximale des différences entre l'index maximal et l'index minimal des intervalles contenus dans chaque ligne et chaque colonne de cette table d'index. Dans son article sur la MDSQ [Vai93], Vaishampayan propose d'allouer les index suivant le critère de la minimisation de la dispersion. Pour cela, il répartit les index autour de la diagonale principale de la table. Cette disposition a le double avantage de permettre effectivement la minimisation de la dispersion et de fournir un moyen simple de régler la redondance. Celle-ci est contrôlée par le paramètre d, où $2d + 1$ est le nombre de diagonales couvertes dans la table d'index. Sur la figure 1.12, d est égal à 2. Ensuite, Vaishampayan propose un ordre de balayage pour répartir les index sur les emplacements occupés de la table d'index. Il conjecture que la dispersion est alors minimale. Par la suite, Berger-Wolf et Reingold [BWR99] [BWR] proposent un algorithme d'allocation de table d'index optimal. Cependant, le gain par rapport à la méthode originale de Vaishampayan est minime. D'autres travaux ont été effectués sur ce sujet [CEL02].

Une fois la table d'index fixée se pose la question du choix des niveaux de reconstruction pour les trois quantificateurs inverses. Vaishampayan [Vai93] propose un schéma d'optimisation global similaire à l'algorithme de Lloyd-Max, qui calcule à la fois les

intervalles de quantification et les valeurs de reconstruction. De façon générale, pour des contraintes de débit données, donc pour des valeurs de M_1 et de M_2 données, et pour des bornes supérieures D_{max_1} et D_{max_2} fixées pour les distorsions des signaux transmis respectivement sur les canaux 1 et 2, un quantificateur MDSQ optimal sera défini par la partition A, les ensembles $\hat{X}^0$, $\hat{X}^1$, et $\hat{X}^2$ tels que la distorsion $E(D_0(X,\hat{X}^0)) = \sum_{(i,j)\in\mathcal{C}} \int_{A_{ij}} D_0(x,\hat{x}^0_{ij})p(x)dx$ soit minimale sous contraintes de $E(D_1(X,\hat{X}^1)) = \sum_{i\in I_1} \int_{A^1_i} D_1(x,\hat{x}^1_i)p(x)dx \leq D_{max_1}$ et $E(D_2(X,\hat{X}^2)) = \sum_{j\in I_2} \int_{A^2_j} D_2(x,\hat{x}^2_j)p(x)dx \leq D_{max_2}$. Le terme $p(x)$ représente la fonction densité de probabilité de la source. Un codage entropique des index peut en outre être pris en compte dans la conception du quantificateur [VD94].

Plusieurs applications de la MDSQ ont été proposées. Dans [YV95], Yang et Vaishampayan montrent que la MDSQ est particulièrement bien adaptée à la transmission de données sur des canaux à évanouissement de Rayleigh et permet de réduire les délais par rapport aux autres techniques existantes. La MDSQ est également appliquée au codage d'image dans [SC98], [SRVN00] et au codage vidéo dans [VJ99]. Le problème du codage prédictif pour la vidéo est considéré spécifiquement dans [NZ01].

L'analyse des performances de la MDSQ a été abordée dans [Vai93]. Les performances asymptotiques de la MDSQ sont étudiées en détail dans [VB98]. L'application de la MDSQ au codage par transformée est considérée dans [BV94] et [BV97]. Dans ce dernier article, il est montré que sous l'hypothèse de haute résolution, les distorsions centrales D_0 et latérales $D_1 = D_2$ d'une MDSQ équilibrée avec $M_1 = M_2 = 2^q$ sont données par

$$\begin{aligned} D_0 &= C\sigma^2 2^{-2q(1+\frac{1}{n})}, \\ D_1 &= S\sigma^2 2^{-2q(1-\frac{1}{n})}, \end{aligned} \tag{1.27}$$

avec $d = 2^{\frac{q}{n}}$, σ^2 la variance de la source et avec C et S déterminés par la fonction de densité de probabilité de la source et par la méthode de codage. Ce résultat est ensuite généralisé au codage en sous-bandes. Dans [VBC98], il est souligné que les performances de la MDSQ sont nettement inférieures à la borne théorique d'Ozarow [Oza80]. Les auteurs suggèrent l'utilisation de la quantification codée en treillis (TCQ) pour réduire cet écart.

1.3.2.2 MDSQ et TCQ

La quantification codée en treillis (TCQ) [MF90] consiste à appliquer le principe de la modulation codée en treillis (*trellis coded modulation, TCM*) au codage de source. Cette technique apporte une réduction de distorsion par rapport à un quantificateur scalaire classique. De même, la TCQ permet d'améliorer les performances de la MDSQ. Une mise en oeuvre de cette idée a été proposée dans [JT99], puis dans [WO00] et [LAK00].

Soit une source de mots de codes S que l'on souhaite coder en deux descriptions 1 et 2 de débits respectifs R_1 et R_2. Suivant le principe de la TCQ, les mots de codes de chacune des deux descriptions prennent leurs valeurs respectivement dans des alphabets

de tailles $M_1 = 2^{R_1+1}$ et $M_2 = 2^{R_2+1}$. Une table d'index MDSQ de taille $M_1 \times M_2$ permet d'associer les deux séquences S_1 et S_2 à la séquence S. On remarque que les mots de codes de S prennent leurs valeurs dans un alphabet de taille $M_1 \times M_2 = 2^{R_1+R_2+2}$. La succession de mots de codes de S_1 correspond à une suite d'états dans le treillis T_1. La séquence S_1 est codée par une suite de transitions dans T_1. De même, la séquence S_2 est codée par une suite de transitions dans un treillis T_2. Par conséquent, la séquence S est codée par des paires de transitions dans les treillis T_1 et T_2. Chaque paire de transition dans ces deux treillis identifie une transition dans le treillis produit $T = T_1 \otimes T_2$. Lors du codage, on cherche un chemin optimal dans le treillis T en fonction des performances souhaitées en décodage central et en décodage latéral. Plus précisément, on cherche à minimiser la distorsion centrale D sous la contrainte des distorsions latérales D_1 et D_2. La métrique à minimiser est de la forme $J = D + \lambda_1 D_1 + \lambda_2 D_2$. Le problème de l'allocation de la table d'index est le même qu'en MDSQ classique.

1.3.2.3 Quantification vectorielle à descriptions multiples

De même que pour la TCQ, la MDSQ peut s'étendre à la quantification vectorielle [BVL94]. On parle alors de quantification vectorielle à descriptions multiples (MDVQ) ou de quantification vectorielle sur réseau de points régulier à descriptions multiples (MDLVQ pour *Multiple description lattice vector quantization*). Ce dernier cas a été traité par Vaishampayan, Sloane et Servetto [SVS99] [VSS01], pour deux descriptions équilibrées. Que ce soit pour le codeur ou bien pour les trois décodeurs, l'espace de représentation d'un vecteur de longueur N est $\mathbb{R}^N$. Le principe de la MDLVQ est similaire à celui de la MDSQ. L'idée consiste à étiqueter chaque point du réseau régulier par une paire d'indices correspondants respectivement à chacune des deux descriptions. Toute la difficulté de la MDLVQ réside dans cet étiquetage. En effet, comme pour la MDSQ, le nombre de combinaisons est trop important pour envisager une recherche exhaustive. De plus les heuristiques utilisées pour la MDSQ reposent sur un ordre de balayage, qui ne peut pas être défini dans un espace de dimension N. Dans [VSS01], les auteurs surmontent cette difficulté en exploitant les symétries des réseaux de points. Tout d'abord, un réseau $\Lambda \subset \mathbb{R}^N$ et un "sous-réseau" $\Lambda' \subset \Lambda$ sont choisis. Le premier fixe la résolution du décodeur central et le second détermine les points de reconstruction pour les décodeurs latéraux. Λ' doit être identique à Λ à une rotation et à un facteur d'échelle près. L'étiquetage optimal $l : \Lambda \rightarrow \Lambda' \times \Lambda'$ est relativement simple, dans la mesure où il peut être défini sur une cellule élémentaire, puis étendu à tout l'espace en appliquant les symétries appropriées.

Considérons l'exemple de la figure 1.13. Sur la figure 1.13-(a), l'ensemble des points constituent le réseau Λ. Le réseau Λ' est quant à lui indiqué par les points noirs. La figure 1.13-(b) présente un exemple d'étiquetage optimal. Si par exemple la paire ab a été codée, la réception du premier indice entraînera le décodage du point A, alors que la réception du second indice entraînera le décodage du point B.

A l'origine, la MDLVQ ne permettait que la création de descriptions équilibrées. Cette limitation a pu être levée par la suite [DSV00] [DSV02]. La MDVQ et la MDLVQ ont également été considérées dans [FE99], [FDZ00], [KGK00] et [Car01].

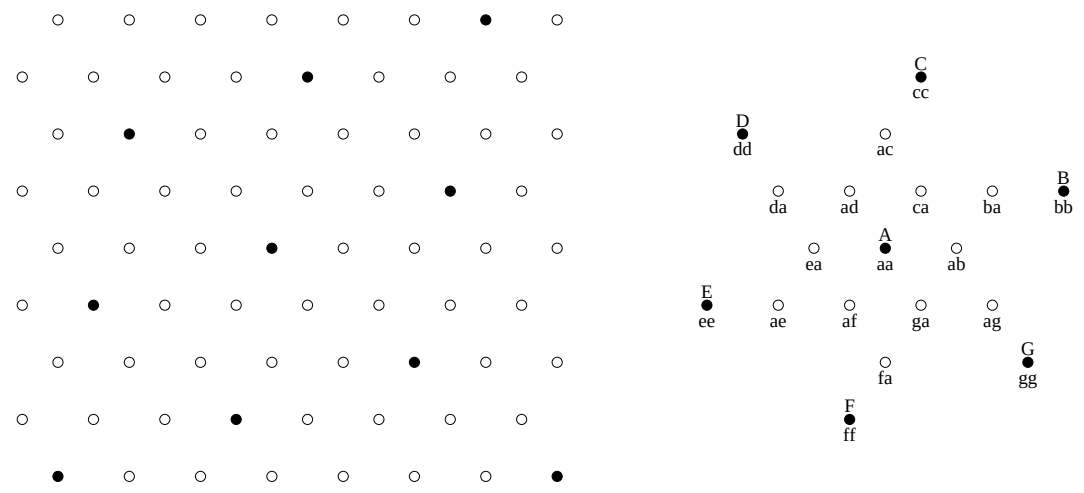


FIG. 1.13 – *(a) Exemple de réseau de points régulier pour la MDLVQ, (b) Exemple d'étiquetage optimal.*

1.3.3 Codage MD basé codes de canal

Initialement, le codage par descriptions multiples a été considéré comme une technique de codage conjoint source/canal. Ainsi, toutes les techniques que nous avons vu jusqu'à présent interviennent au niveau du codage de source. Récemment, Une technique de codage par descriptions multiples qui sépare le codage de source de la formation des descriptions a été proposée par Mohr, Riskin et Ladner [MRL99b] [MRL99a] et Puri et Ramchandran [PR99]. Il s'agit de la *protection inégale aux pertes*, ou *unequal loss protection, ULP*. Sur la figure 1.3, cette technique s'insère au niveau du codage de canal.

Cette technique trouve son origine dans le principe de la *protection inégale aux erreurs*, ou *unequal error protection, UEP*. Cette dernière a été développée pour assurer une protection efficace des flux de données progressifs ou "scalables". De plus en plus dans le développement de techniques de compression, en particulier pour les données images [MGBB00] et vidéo [ISO02], on cherche à inclure la caractéristique de progressivité, ou "scalabilité". Cette caractéristique permet d'adapter aisément le débit de transmission des données à l'état du canal ou aux possibilités du récepteur. Les flux de données ainsi créés sont dit emboîtés. Un tel flux est ordonné. C'est à dire qu'un paquet p_i ne peut être exploité que si les $i-1$ paquets $p_1 \dots p_{i-1}$ précédents ont été reçus. Il est par conséquent primordial de recevoir les paquets dans l'ordre. La perte d'un paquet p_i empêche le décodage de tous les paquets suivants. Les techniques de codage de canal classiques assurent une même protection sur l'ensemble des données. La protection inégale aux erreurs a été introduite pour traiter spécifiquement le cas des flux emboîtés. En assurant une meilleure protection du début du flux de données que de la fin, de meilleurs compromis débit/robustesse peuvent être atteints [ABE+96].

La protection inégale aux pertes repose sur l'association d'un flux de données

emboîté avec un code correcteur d'erreur (*forward error correction, FEC*). Il faut bien remarquer que la caractéristique de progressivité est indispensable au fonctionnement de cette technique. Un nombre quelconque N de descriptions peut être genéré par cette méthode.

Soit un flux de données emboîté F. A partir de F, on cherche à créer N descriptions telles que la réception de n'importe lesquelles k d'entre elles conduisent à la distorsion $d(k)$. F est partitionné en N blocs, tels que la réception des k premiers blocs conduit à la distorsion recherchée $d(k)$. Les k premiers blocs doivent pouvoir être décodés lorsque k descriptions sont reçues. Autrement dit, un bloc k doit pouvoir être décodé si k descriptions ou plus sont reçues. Chaque bloc k est découpé en k sous blocs. Un code correcteur est appliqué sur chacun de ces k sous blocs de façon à obtenir N sous-blocs correspondants au bloc k et tels que la réception de k de ces sous-blocs suffise à décoder le bloc k. Les codes de Reed-Solomon (RS) ont la propriété de distance maximale et peuvent donc être utilisés dans ce contexte. Finalement, pour chaque bloc de données k, N sous-blocs sont créés et répartis dans les N descriptions. La structure des descriptions ainsi formées est résumée sur la figure 1.14 pour $N = 4$ descriptions. Le problème du réglage des valeurs de $d(k)$ pour chaque k consiste à maximiser l'espérance de la distorsion en fonction de la probabilité de perte des descriptions. Ce problème est traité de manière optimale dans [MRL99b]. Dans [PR99], les auteurs proposent un algorithme rapide et presque optimal.

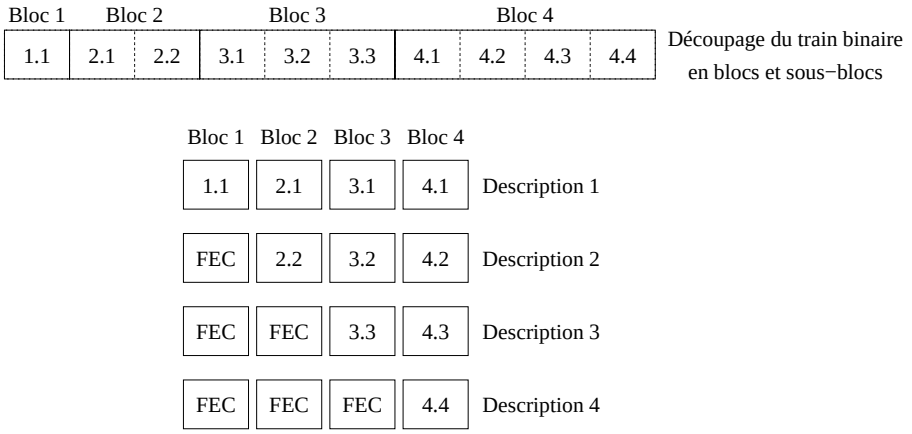


FIG. 1.14 – *Construction de 4 descriptions à partir d'un train binaire progressif.*

Dans [MRL99a], les auteurs soulignent les avantages du codage par descriptions multiples basé sur la protection inégale aux pertes. Parmi ceux-ci, on trouve :

- la séparation source/canal : le retour a un tel schéma permet un codage de source efficace et une mise à jour aisée de ce codage de source. De plus, l'ajout de redondance est facile à quantifier, puisqu'il est indépendant de l'étape de compression. La séparation source/canal implique une conception simple et modulaire du schéma de codage.
- Des descriptions parfaitement équilibrées : chaque description à la même impor-

tance que les autres. La distorsion dépend uniquement du nombre de descriptions reçues.

- La construction optimale des descriptions: la répartition de l'information dans les descriptions peut être effectuée de manière optimale en fonction des conditions de transmission.

Le codage par descriptions multiples basé sur la protection inégale aux pertes a été approfondi à la fois d'un point de vue théorique et d'un point de vue pratique dans [PPR02a] et [PPR02b]. Plusieurs applications de cette technique de codage ont été proposées pour la transmission d'images fixes [PR99] [RV00] [MMR99] [SAR00]. Dans [VFE00], le schéma de codage est enrichi par l'inclusion d'information d'état dans les données compressées. Cette information est alors exploitée par le décodeur. Le principe de protection inégale peut être amélioré en tenant compte de l'impact visuel induit par les pertes. Dans [MR00], la présence de régions d'intérêts (ROI) dans l'image est prise en compte dans le réglage de la protection. Dans [CC00], une classification est effectuée préalablement à l'allocation de redondance, de manière à protéger en priorité les zones spatiales de l'image qui ont le contenu visuel le plus important. Finalement, dans [ZMG02], une application au codage vidéo est proposée.

Pour conclure, nous remarquons que si le problème de l'allocation de redondance et de la répartition de l'information dans les descriptions a été traité, le choix du nombre optimal de descriptions, en revanche, ne l'a pas été.

1.4 Codage vidéo à descriptions multiples

L'utilisation du codage par descriptions multiples plutôt que des mécanismes de retransmission ou des codes de canal appliqués sur des longs blocs de données est souvent liées à des contraintes de délai. La diffusion de vidéo sur des réseaux de type internet est une application en plein développement. Les contraintes de délais qui doivent alors être respectées conduisent naturellement à concevoir des codeurs vidéo à descriptions multiples. La difficulté dans la conception de tels codeurs réside dans l'aspect prédictif de la compression vidéo. On parle de *drift* lorsque le codeur et le décodeur sont désynchronisés lors de la transmission de vidéo. Un certain nombres de schémas de codage vidéo à description multiples qui ignorent ou qui contournent le problème du drift ont été proposés. L'impact du drift a été évalué par Hénocq [Hén02], dans le cadre d'un codeur vidéo basé sur les transformations sur bases de fonctions redondantes. Ces travaux montrent la nécessité de traiter le problème du drift. Une solution basée sur le codage et la transmission des erreurs de prédictions correspondant aux différentes situations de décodage est alors proposée. Notons que ce problème du drift est également présent en codage vidéo "scalable".

Nous présentons dans cette section quelques exemples de solutions qui ont été proposées pour le codage vidéo à descriptions multiples, en commençant par celles qui traitent au mieux le problème du drift.

1.4.1 Codage vidéo à trois boucles de prédiction

Un des premiers schéma de codage vidéo à descriptions multiples proposé est basé sur l'appariements de doublets [RPJ+99] [RJW+99]. Le point clé lors du développement d'applications vidéos basées descriptions multiples s'appuyant sur de la compensation de mouvement réside dans la recherche d'une erreur de prédiction adaptée. Une application directe des schémas vidéos mono-couche aboutira à une erreur de prédiction pour chaque image de chaque description. Si les deux canaux sont reçus, la meilleure prédiction possible sera obtenue avec les informations provenant de chaque description. Les problèmes se posent lorsque l'un des canaux est perdu. En effet, dans ce cas l'erreur de prédiction de la description perdue ne sera pas disponible et il se créera un décalage dans la reconstruction. Ces erreurs ne seront corrigées que par des macro-blocs INTRA. On peut donc considérer que dans un tel schéma deux types de distorsion existent: la distorsion due à la quantification de l'erreur de prédiction et la distorsion due aux décalages possibles entre les boucles de prédiction du codeur et du décodeur. Le schéma de la figure (1.15) proposé dans [RPJ+99] et [RJW+99] gère ces deux distorsions. On peut considérer que F représente l'erreur de prédiction et $G1 - F$ et $G2 - F$ les erreurs de décalage. On peut remarquer que dans le schéma seules deux boucles sont représentées, la troisième étant obtenue par symétrie. Les trois boucles permettent de simuler les trois scénarios possibles au décodeur: les deux descriptions sont reçues ou l'une des descriptions est perdue. Le codeur aura donc trois buffers d'images contenant la dernière image reconstruite pour chaque boucle. Pour chaque macro-blocs X le codeur génère trois prédictions $Pred_i (i = 0, 1, 2)$. Une DCT est appliquée sur les macro-blocs dans la boucle centrale et le résultat est quantifié. On forme ensuite $\frac{N_{coeff}}{2}$ paires de coefficients DCT. Chaque paire est ensuite soumise à une transformation de corrélation (Pairwise Correlating Transform, PCT) de paramètre β et le résultats est séparé en deux ensembles.

La PCT introduit une corrélation contrôlée entre les deux flux de coefficients. Si A et B sont les coefficients DCT originaux, la sortie de la PCT donne C et D où :

$$\begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2\beta}} & \sqrt{\frac{\beta}{2}} \\ -\frac{1}{\sqrt{2\beta}} & \sqrt{\frac{\beta}{2}} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \tag{1.28}$$

Au décodeur, si seule une description est disponible, les coefficients de l'autre description sont estimés à partir de cette corrélation en utilisant une prédiction linéaire.

Pour réduire les futurs décalages entre les deux prédictions, les informations résiduelles G_1 et G_2 sont utilisées dans les boucles latérales. Toutefois, plutôt que de coder complètement G_1 et G_2, on essaye d'extraire les parties les plus importantes du signal de manière à pouvoir les représenter par 32 coefficients. Pour cela un codeur basé transformation généralisée, introduit dans [WOR98], est utilisé. Seule l'information orthogonale à l'information de la boucle centrale est envoyée. Ainsi PCT_p dans la figure (1.15) reçoit l'information du canal 1 de la part de la boucle centrale, calcule la meilleure prédiction possible avec cette information et génère en sortie l'information G_1 n'ayant pas été représentée par F dans le sous espace orthogonal à l'information du canal 1.

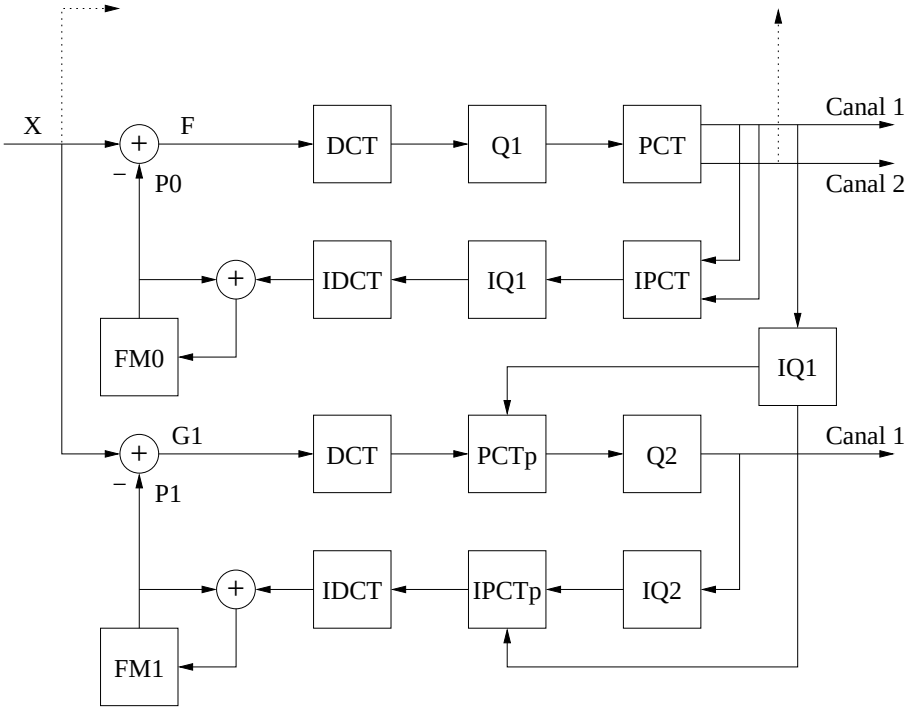


FIG. 1.15 – *Codage vidéo par descriptions multiples basé sur 3 boucles de prédiction. Deux boucles sont représentées, la troisième se déduit par symétrie.*

Ces 32 coefficients de la boucle latérale 1 sont ensuite ajoutés aux 32 coefficients du canal 1. On utilise une méthode similaire pour la voie 2.

Pour l'allocation de la redondance, on contrôle la redondance sur F en faisant varier le paramètre de la PCT. Pour de faibles niveaux de résistance aux pertes, on alloue de la redondance uniquement à F. Pour permettre plus de résistance un compromis doit être trouvé entre redondance accordée à F et redondance accordée aux composantes G_i. On peut noter que l'information contenue dans les paramètres G_i est de la redondance pure puisqu'elle sera inutile au décodeur si les deux descriptions sont reçues. Cette approche à été introduite dans un codeur H.263+. Les auteurs de [RPJ+99] et [RJW+99] ont proposé par la suite un schéma de codage mieux adapté aux fortes redondances [RJWO01]. Le principe de codage vidéo à trois boucles de prédiction est réutilisé dans [TZ01], [LAM02] et [WL01].

1.4.2 Codage vidéo à deux boucles de prédiction

Dans [VJ99], Vaishampayan et John proposent un schéma de codage prédictif à descriptions multiples où deux boucles de prédiction sont suffisantes. La technique est d'abord développée dans le cas de la quantification prédictive (*differential pulse code modulation, DPCM*), puis est étendue au codage vidéo. Cette méthode est basée

sur la MDSQ. Les auteurs montrent qu'en concevant les intervalles de quantification et la table d'index d'une manière spécifique, il est possible de définir un schéma ou deux boucles de prédiction sont appliquées indépendamment sur chacune des deux descriptions. Le décodage d'une seule description ne pose alors aucun problème. Lorsque les deux descriptions sont reçues, elles ont la propriété de se raffiner mutuellement. Un principe similaire est exploité dans [NZ01].

1.4.3 Autres techniques

Dans [BZLS01], les auteurs proposent un schéma où les deux descriptions sont séparées en deux parties : une partie strictement redondante, et une partie non redondante qui raffine la première partie. Le principe se rapproche de la transformée polyphase. Seule la partie redondante est utilisée comme image de référence pour la compensation de mouvement. Par conséquent, le problème du drift est éliminé, au prix d'une légère perte en compression.

Dans [Apo00] et [AW01], un schéma de codage vidéo assimilable à une transformée polyphase temporelle est proposé. Les images paires et impaires de la séquence sont réparties entre les deux descriptions. La compensation de mouvement est effectuée indépendamment pour chaque description, ce qui élimine le problème du drift. En cas de perte d'une description, la séquence vidéo est sous-échantillonnée temporellement, ce qui reste visuellement acceptable. De plus, les images perdues peuvent être estimées ou interpolées à partir de celles qui sont correctement reçues.

Le principe de la transformée polyphase est également utilisé dans [GSK01], mais dans le domaine spatial, avant l'étape de codage vidéo, qui est alors appliquée indépendamment sur les deux descriptions. Dans ce cas, la création des descriptions et le codage vidéo sont complètement découplés, ce qui élimine tout problème de drift. Un tel schéma n'est cependant pas forcément efficace d'un point de vue compression [CTK01].

L'approche proposée dans [RR00] est radicalement différente. Sachant l'existence du drift, le problème de la prédiction pour chaque décodeur est reformulé par une estimation optimale de l'échantillon courant sachant les informations disponibles au décodeur. Des simulations sur des sources de Markov, ainsi que sur des vrais signaux vidéo montrent la validité de l'approche.

Enfin dans [CSO01], un schéma de codage à deux descriptions déséquilibrées est proposé. La séquence vidéo est codée en deux versions, l'une à haute résolution et l'autre, déduite de la première, à basse résolution. La version basse résolution constitue une redondance pure. Elle n'est utilisée qu'en cas de perte dans la version haute résolution. Une optimisation débit/distorsion permet de minimiser la distorsion attendue en fonction des caractéristiques de pertes du canal. En cas de perte, un effet de drift apparaît. Cependant, dans ce contexte de codage par descriptions multiples déséquilibré, son impact est limité. La version basse résolution étant codée à très bas débit, il y a peu de marge de manoeuvre pour la transmission d'une information adjacente permettant de compenser le drift.

1.4.4 Codage vidéo basé sous-bandes 3D

Nous concluons par une catégorie particulière de codeurs vidéo, dont le développement est très actif actuellement. Il s'agit des codeurs vidéo basé transformée en ondelettes 3D, ou 2D+t. Dans ces schémas, la prédiction est remplacée par un filtrage temporel, qui peut lui aussi être compensé en mouvement [SJJ99]. Cette approche du codage vidéo a l'avantage de permettre la création de flux finement "scalables".

Dans [SN99], une extension du schéma de codage par descriptions multiples d'image fixe basé sur SPIHT de [SRVN00] à la vidéo est proposée. Dans [TZ99], Les sous bandes spatio-temporelles sont découpées en blocs. Ces blocs sont ensuite répartis entre les différentes descriptions, de façon à ce que chaque description contienne de l'information sur toutes les zones spatiales des images. Ces deux schémas ont été développés de manière à s'inscrire dans un schéma global de communication. Dans [SN99], une modification du protocole de transport standard internet (TCP) est associée à la conception du codeur vidéo à descriptions multiples.

Enfin dans [PAB02], un algorithme d'allocation global permet d'obtenir un compromis optimal entre efficacité de compression et robustesse. Les sous bandes spatio-temporelles sont dupliquées pour former les deux descriptions. L'algorithme d'allocation à alors la charge de partager le débit entre les sous bande des deux descriptions en fonction des caractéristiques d'erreur du canal de manière à maximiser la qualité de reconstruction.

Les trois exemples présentés n'utilisent pas de compensation de mouvement. L'inclusion de cette dernière dans un codeur vidéo par descriptions multiples basé sous-bandes 3D reste un problème ouvert.

1.5 Conclusion

Dans ce chapitre, nous avons présenté le codage par descriptions multiples. Après avoir introduit le sujet et donné les définitions nécessaires, des résultats théoriques sur le codage à deux descriptions et sur le codage à plus de deux descriptions ont été présentés. Ensuite, un état de l'art des techniques de codage par descriptions multiples a été proposé. Ces techniques ont été classées en trois grandes catégories : les méthodes basées transformation, quantification et codage de canal. Enfin, des exemples d'applications pratiques au codage vidéo ont été présentés.

Initialement, le codage par descriptions multiples a été proposé pour la transmission de données sur plusieurs canaux distincts et non corrélés. Il s'applique aussi naturellement aux transmissions sur des réseaux par paquets où les pertes de paquets sont aléatoires et indépendantes. Dans tous les cas, l'hypothèse de l'erreur résiduelle nulle doit être respectée. A partir du moment où des techniques pratiques de codage par descriptions multiples ont été proposées, il a été possible de vérifier expérimentalement l'intérêt de ces techniques.

Dans [SOPJ00], le codage par descriptions multiples est comparé au codage progressif, ou codage en couches, dans le cas de la transmission sur un réseau de type internet. Le codage progressif permet d'adapter le débit de transmission des données en fonction

des caractéristiques courantes du réseau. Cependant, si la première couche est perdue, toutes les autres couches sont inexploitables. Lorsqu'il est nécessaire de respecter à la fois des contraintes de fiabilité et de délai, cet article montre la supériorité du codage par descriptions multiples pour une large gamme de conditions de transmissions, et ce malgré la redondance plus importante introduite par ce dernier. Un résultat similaire est obtenu dans [ASPEF01], où le codage classique à une seule description est comparé au codage par descriptions multiples dans le cadre d'un réseau sujet à des congestions. Une distorsion moyenne plus faible est obtenue en remplaçant le codage à simple description par le codage à deux descriptions. Dans [YV95], Yang et Vaishampayan considèrent la transmission de données sur des canaux à évanouissement de Rayleigh. Classiquement, pour la transmission sur de tels canaux, les données sont entrelacées, de manière à décorréler les pertes. Cet entrelacement introduit un délai dans la transmission. Yang et Vaishampayan montrent que le codage par descriptions multiples permet d'obtenir des performances identiques au techniques classiques, tout en réduisant le délai.

D'une manière générale, le codage par descriptions multiples est parfaitement adapté à la prise en compte des effacements lors de la transmission de données sous contrainte de délai. En revanche, il n'est pas adapté aux transmissions sujettes aux erreurs. Dans [CKS01], les performances du codage par descriptions multiples sont évaluées dans le cadre de la transmission sur des canaux à erreurs bits. Des techniques plus classiques où le codage de source et le codage de canal sont optimisés conjointement se révèlent être plus efficaces.

Si l'intérêt du codage par descriptions multiples a pu être montré, les comparaisons des performances des différentes techniques sont plus rares. Pour les applications au codage d'images, le codage à deux descriptions basé transformée polyphase et quantification sélective apparaît comme la technique la plus efficace d'une part en termes de compromis distorsion centrale/distorsion latérale et d'autre part en termes de complexité. Plus généralement, le choix d'une technique dépend du contexte d'application.

Les techniques basées transformations ont pour avantage de permettre une exploitation efficace de la redondance. Elles peuvent même, comme c'est le cas des transformations sur bases de fonctions redondantes, exploiter cette redondance pour corriger une partie de l'erreur de quantification. Elles ont en revanche l'inconvénient d'une certaine complexité. De plus, le réglage de la redondance peut s'avérer délicat. En effet, dans le cas de bancs de filtres à descriptions multiples, par exemple, il faut synthétiser un banc de filtres par niveau de redondance. Il paraît difficile dans ce cas d'adapter dynamiquement la redondance en fonction des caractéristiques du canal. Même si l'on peut imaginer des solutions permettant d'adapter la redondance, il n'en reste pas moins qu'il faut recoder complètement les données à chaque modification. En effet, les transformations se situent au début de la chaîne de codage. Notons tout de même que la transformée polyphase constitue une exception. C'est une transformation triviale, et le réglage de la redondance s'effectue au niveau de la quantification. En revanche, lorsque toutes les descriptions sont reçues, la partie redondante n'est absolument pas exploitée.

La conception de techniques de codage par descriptions multiples basées quantification est rendue délicate par le problème de la conception des tables d'index dans le cas de la quantification scalaire et par le problème de l'étiquetage dans le cas de la

quantification vectorielle. Ces techniques permettent cependant d'atteindre des performances intéressantes. nous retiendrons la quantification scalaire à descriptions multiples (MDSQ) pour ses performances, sa facilité de réglage de la redondance grâce à un paramètre unique et la possibilité de l'insérer aisément dans un modéle bayésien de la chaîne de codage, comme cela sera vu au chapitre 5.

Finalement, le codage par descriptions multiples basé sur la protection inégale aux pertes à la caractéristique de séparer le codage de source du codage de canal, ce qui peut être vu comme un avantage ou bien un inconvénient, suivant l'application. En effet, cela procure une certaine souplesse dans le réglage de la redondance, qui peut être effectué sans recoder les données. De plus, cela permet de créer un nombre quelconque de descriptions. En revanche, contrairement à un schéma de codage conjoint, cela ne permet pas d'exploiter au mieux la redondance. Lorsque toutes les descriptions sont reçues, la partie redondante n'est absolument pas exploitée. Notons enfin que cette technique ne s'applique que sur des sources codées progressivement.

Chapitre 2

Codage progressif par descriptions multiples

2.1 Introduction

La section 1 a mis en évidence les avantages et inconvénients des différentes techniques de codage par descriptions multiples, indépendamment de leur contexte d'exploitation. L'utilisation de ces techniques dans un cadre applicatif particulier peut s'avérer problématique. Par exemple, l'utilisation du codage par descriptions multiples pour le codage vidéo est rendue délicate par la présence de la compensation de mouvement, qui doit être synchronisée entre le codeur et le décodeur (section 1.4). Nous considérons à présent l'application des techniques de codage par descriptions multiples au codage d'images fixes. Le but de ce chapitre est de mettre en avant les difficultés posées par le codage d'image à descriptions multiples et de proposer des solutions adaptées.

Plutôt que de développer complètement un codeur basé descriptions multiples, nous cherchons à déterminer si le codage par descriptions multiples peut être intégré efficacement dans un codeur existant. Pour effectuer cette étude, nous utilisons le codeur d'images fixes JPEG2000 comme codeur de référence. Un tel codeur n'a pas pour seul intérêt ses performances en compression. Il propose également un certain nombre de fonctionnalités [MGBB00], telles que :

- transmission progressive par qualité, résolution ou composante.
- Compression avec ou sans perte.
- Accès aléatoire au train binaire, décodage d'une partie seulement de l'image.
- Traitement dans le domaine compressé (par exemple, rotation ou rognage).
- Codage prioritaire de régions d'intérêts.

Parmi toutes ces fonctionnalités, la progressivité fait partie de celles qui ont motivé la création du standard JPEG2000. En effet, la progressivité permet, à partir d'un unique train binaire compressé, la transmission et la visualisation des images sur des canaux et des terminaux hétérogènes. C'est donc une fonctionnalité particulièrement bien adaptée aux besoins actuels. Ce besoin d'adaptation aux canaux et aux terminaux

se retrouve pour le codage vidéo. Ainsi, les standards de codage vidéo récents, tels que H263+ [CEGK98] ou MPEG4 [ISO02] incluent la fonctionnalité de "scalabilité".

Le fonctionnement de JPEG2000 est détaillé en annexe A. Nous le résumons ici par le schéma bloc de la figure 2.1. L'élément clé du codeur JPEG2000 est l'algorithme EBCOT (Embedded Block Coding with Optimized Truncation), qui se divise en deux parties. Dans la première partie, chaque sous-bande quantifiée est découpée en blocs, appelés *code-blocks*. Ces code-blocks sont codés indépendamment les uns des autres, ce qui permet le décodage partiel de l'image. Le train de bit généré pour chaque code-block est dit emboîté, ou progressif. C'est à dire qu'il peut être tronqué à différents débits, tout en restant optimal du point de vue du rapport débit/distorsion. Pour obtenir cette progressivité, un codage par plan de bits est adopté. C'est à dire que les code-blocks sont des tableaux d'entiers en représentation signe-amplitude, où les amplitudes sont ordonnées par plans de bits, des bits les plus signifiants aux moins signifiants. Pour atteindre une progressivité encore plus fine, les plans de bits sont ré-ordonnés en trois *sous-plans de bits*, de manière à coder en priorité les bits qui apportent le plus d'information. Ainsi, chaque plan de bits est parcouru trois fois, chaque passe correspondant respectivement

1. aux entiers non signifiants prédits comme allant devenir signifiants,
2. aux entiers qui ont déjà été trouvés signifiants,
3. aux entiers non signifiants prédits comme allant rester non signifiants.

Un entier est dit signifiant lorsqu'au moins un bit correspondant à cet entier est égal à 1 dans les plans de bits précédents. Les bits ainsi ordonnés sont codés par un codeur arithmétique binaire contextuel. Le contexte de chaque bit est défini par la *signifiance* et le signe de chacun des entiers voisins spatialement. Il n'y a aucune exploitation de dépendances inter-code-block ou inter sous-bande. Le train binaire généré pour chaque code-block peut être tronqué toutes les passes. Pour chaque passe, un incrément de distorsion est calculé et conservé. Ainsi une courbe débit/distorsion est définie pour chaque code-block. La deuxième partie d'EBCOT utilise cette information pour créer le train binaire final. Une procédure d'optimisation débit/distorsion permet d'avoir un train binaire progressif optimal. Ce train binaire, découpé en *paquets EBCOT* et organisé en *couches de qualités*, peut être ré-ordonné en fonction du type de progressivité désiré. En résumé, les fonctionnalités principales de JPEG2000 reposent sur l'utilisation

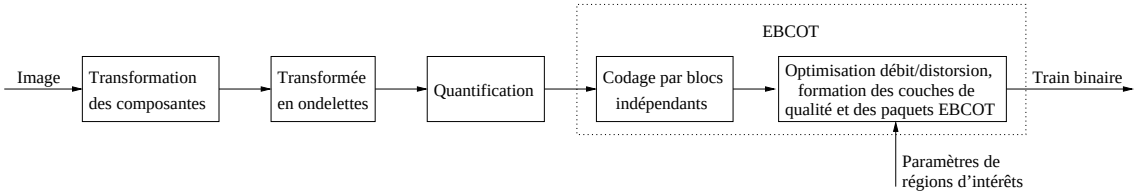


FIG. 2.1 – *Schéma bloc du codeur JPEG2000*

de la transformée en ondelettes (progressivité en résolution), du codage par plans de bits (progressivité en qualité), du découpage en code-block (accès aléatoire aux régions

spatiales de l'image) et d'une organisation flexible du train binaire (traitement dans le domaine compressé).

Parmi les techniques de codage par descriptions multiples étudiées dans le chapitre 1, nous avons retenu la quantification scalaire à descriptions multiples [Vai93] et la transformée polyphase couplée à la quantification sélective [JO99]. Nous considérons dans ce chapitre l'utilisation de ces deux techniques pour la création d'un codeur d'image basé JPEG2000 à deux descriptions. La transformée polyphase s'applique naturellement. La MDSQ, en revanche, doit faire l'objet d'une étude plus approfondie. Dans la section 2.2, nous considérons le problème de la progressivité en dehors du contexte JPEG2000. Puis, dans la section 2.3, nous étudions le codage d'image basé JPEG2000 à deux descriptions.

2.2 Codage progressif à descriptions multiples

2.2.1 Définition

Nous appelons codage progressif le fait de coder une source en un train binaire *successivement raffinable*, suivant la définition donnée dans [EC91]. Un train binaire est dit successivement raffinable si pour tout R, la distorsion obtenue en décodant ce train binaire tronqué au débit R est $D(R)$, où $D(R)$ est la *fonction débit/distorsion* définie par Shannon [Sha59] (section 1.2.1). Dans [EC91], les auteurs montrent dans le cas général *qu'un problème débit/distorsion est successivement raffinable si et seulement si les solutions individuelles des problèmes débit/distorsions forment une chaîne de Markov*. Dans la pratique, cette condition est rarement vérifiée. On retiendra donc comme définition du raffinement successif la possibilité de décoder le train binaire tronqué à un débit quelconque R et d'atteindre une distorsion la plus proche possible de $D(R)$.

La définition du codage à deux descriptions que nous avons donné initialement en section 1 est valide pour un quintuplet $(R_1, R_2, D_0, D_1, D_2)$. Les techniques de codage présentées dans l'état de l'art sont optimisées pour un quintuplet fixé. De même les techniques de codage à plus de deux descriptions sont optimisées pour des débits fixés par description.

Définition : Nous appellons codage progressif à descriptions multiples le fait de coder une source en plusieurs trains binaires progressifs, décodables indépendamment et mutuellement raffinable.

Dans le cas du codage à deux descriptions, cela signifie que si l'on code une source à un débit R_{max} par description, on doit pouvoir décoder la source à partir des deux descriptions tronquées respectivement aux débits R_1 et R_2, avec $R_1 \in [0; R_{max}]$ et $R_2 \in [0; R_{max}]$. La distorsion obtenue doit être la plus proche possible de la limite théorique pour la paire de débits (R_1, R_2) (section 1.2.1).

La définition du codage progressif à descriptions multiples, nous conduit à étendre la terminologie déjà utilisée : nous appelerons

- *décodage latéral*, le décodage progressif d'une seule description ($R_i = 0$, $R_j \in [0; R_{max}]$, $i, j = 1, 2$, $i \neq j$),

- *décodage central*, le décodage progressif à débit égal des deux descriptions ($R_1 = R_2$, $R_1, R_2 \in [0; R_{max}]$),
- *décodage inégal*, le décodage progressif des deux descriptions dans tous les cas excepté le décodage central ($(R_1, R_2) \in [0; R_{max}] \times [0; R_{max}]$, $R_1 \neq R_2$).

2.2.2 Transformée polyphase et quantification sélective

Dans le codeur JPEG2000, la caractéristique de progressivité est obtenue grâce au principe de la transmission par plans de bits. Le codage par descriptions multiples basé sur la transformée polyphase n'influe pas sur les paramètres de quantification. On peut utiliser le codage par plans de bits pour effectuer la quantification sélective, plutôt que de varier les pas de quantification comme dans la technique originale. Le schéma ainsi obtenu est résumé par la figure 2.2. On remarque sur cette figure que la quantification

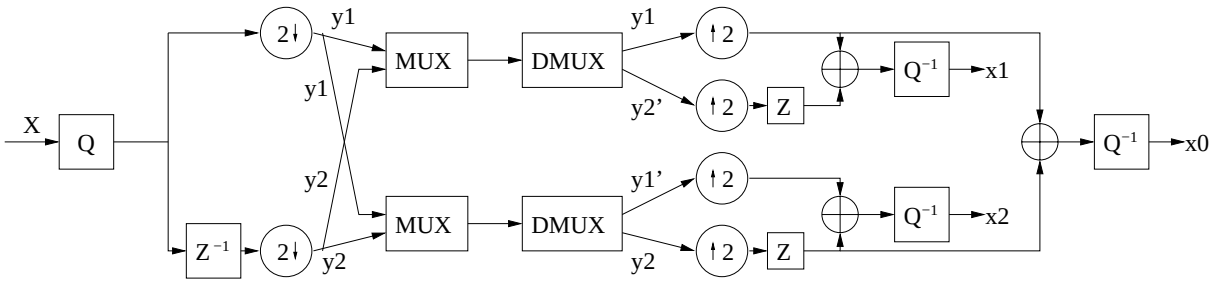


FIG. 2.2 – *Codage par descriptions multiples progressif par transformée polyphase : schéma bloc.*

est effectuée avant la transformée polyphase. Les deux composantes polyphases sont toutes les deux dirigées vers les composants "MUX" (multiplexage), dont le rôle sera alors de combiner ces données et de régler la redondance. Ce réglage s'effectue en contrôlant le nombre de plans de bits transmis pour chacune des deux composantes polyphases. Une fois ce réglage établi, les plans de bits contenus dans chacune des deux descriptions doivent être ordonnés de façon à permettre la progressivité aussi bien en décodage central que latéral. Pour cela, nous proposons d'entrelacer les plans de bits comme cela est décrit dans la figure 2.3. Le principe est d'entrelacer les plans de bits proportionnellement au niveau de redondance désiré. Par exemple, pour un rendement de 4/5, pour chaque description, un plan de bit de la composante polyphase quantifiée grossièrement sera inséré tous les 4 plans de bits de l'autre composante. Les plans de bits les plus signifiants (MSB) des deux composantes sont présents dés le début de chacune des descriptions, afin de limiter le risque de n'avoir aucune information pour une composante.

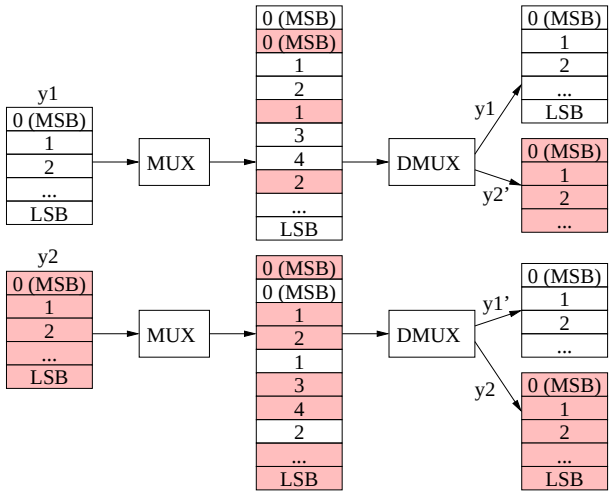


FIG. 2.3 – *Codage par descriptions multiples progressif par transformée polyphase : entrelacement des plans de bits pour un rendement de* 2/3.

2.2.3 Quantification scalaire à descriptions multiples

2.2.3.1 MDUSQ progressive : notations et exemples

Soit un système de codage par quantification scalaire à descriptions multiples composé d'un quantificateur Q et d'une table d'index A avec d diagonales occupées. Le quantificateur convertit une séquence source X à valeurs réelles en une séquence de symboles entiers qui prennent leurs valeurs dans un alphabet fini C de taille M. Chaque symbole est ensuite projeté sur deux ensembles *d'index* qui seront codés et transmis indépendamment sur un canal. Ces deux ensembles d'index prennent leurs valeurs dans les alphabets I et J de tailles respectives M_I et M_J. C est un sous-ensemble de $I \times J$. Cette projection est définie par la table d'index A. Autrement dit, la table d'index A permet de convertir un symbole $c = 1, \dots M$ en deux index $i = 1, \dots M_I$ et $j = 1, \dots M_J$. Chaque symbole c est désigné par ses coordonnées (i, j) dans la table d'index. Le quantificateur Q définit une partition de $\mathbb{R}$ dite *partition centrale*. Les intervalles de cette partition sont désignés par A_{ij}. La table d'index A définit deux partitions de $\mathbb{R}$ dites *partition latérales*. Ces deux partitions ont pour caractéristiques d'être composées d'intervalles disjoints, désignés respectivement par A_i et A_j. Le décodeur est composé de trois quantificateurs inverses qui convertissent les index reçus i et j en valeurs de reconstructions $\hat{x}_{(i,j)}$, $\hat{x}_i$ et $\hat{x}_j$, qui prennent leurs valeurs respectivement dans les dictionnaires $\hat{X}_C = \left\{\hat{x}_{(i,j)}, (i,j) \in C\right\}$, $\hat{X}_I = \{\hat{x}_i, i \in I\}$ et $\hat{X}_J = \{\hat{x}_j, j \in J\}$. Soit $d(x_n, \hat{x}_n)$ la distorsion entre le symbole x et sa valeur reconstruite $\hat{x}_n$. Suivant [Vai93], la distorsion centrale du système de codage MDSQ est donnée par

$$\overline{d_0} = \sum_{(i,j)\in C} \int_{A_{i,j}} d(x, \hat{x}_{i,j}) P(x) dx, \tag{2.1}$$

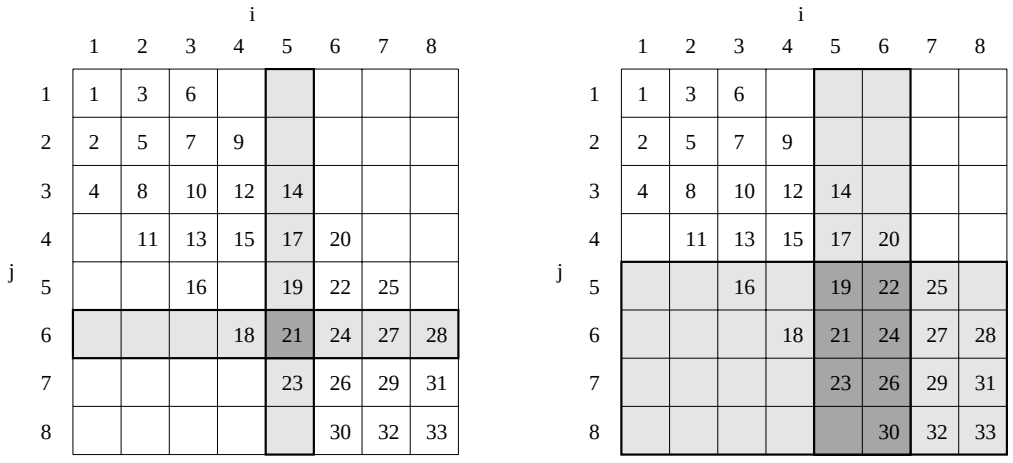
et les distorsions latérales sont données par

$$\overline{d_1} = \sum_{i \in I} \int_{A_i} d(x, \hat{x}_i) P(x) dx, \tag{2.2}$$

et

$$\overline{d_2} = \sum_{j \in J} \int_{A_j} d(x, \hat{x}_j) P(x) dx. \tag{2.3}$$

La figure 2.4-a présente un exemple de table d'index pour $M_I = M_J = 8$ et $C = 33$. La réception des index $i = 5$ et $j = 6$ correspond au décodage du symbole $c = 21$, dont la valeur de reconstruction $\hat{x}_{(5,6)}$ est calculée à partir de l'intervalle $A_{5,6}$. La réception de l'index $i = 5$ uniquement correspond à la valeur de reconstruction $\hat{x}_{i=5}$ calculée à partir de la réunion des intervalles $A_{5,j}, j \in J$.



(a) Décodage MDSQ.

(b) Décodage MDSQ progressif.

FIG. 2.4 – *Tables d'index MDSQ.*

Considérons à présent la transmission progressive indépendante par plans de bits des deux descriptions. Soit q_I et q_J le nombre de plans de bits disponibles pour chacune des deux descriptions ($M_I = 2^{q_I}$, $M_J = 2^{q_J}$). Nous n'avons plus trois scénarios de décodage, comme présenté ci-dessus, mais $(q_I + 1) \times (q_J + 1)$, suivant le nombre de plans de bits reçus pour chacune des deux descriptions. Les bits reçus sur chacune des descriptions permettent de définir un découpage de la table d'index. Les valeurs de reconstructions doivent être calculées pour chacun de ces découpages. La figure 2.4-b présente un exemple où 2 plans de bits ont été reçus pour la première description ($i = 5, 6$) et 1 plan de bit seulement pour la seconde ($j = 5, 6, 7, 8$). Le découpage correspondant de la table d'index est une zone rectangulaire contenant 7 intervalles de

la partition centrale, pour lesquels il faut calculer une valeur de reconstruction optimale. Soient b_i et b_j le nombre de plans de bits reçus pour chacune des descriptions. La distorsion du système de codage MDSQ correspondante est donnée par

$$\overline{d_{b_i b_j}} = \sum_{k=1}^{2^{b_i}} \sum_{l=1}^{2^{b_j}} \left(\sum_{m=1}^{K_i} \sum_{n=1}^{K_j} \int_{(\alpha,\beta)\in C} d(x, \hat{x}_{k,l}) P(x) dx \right), \tag{2.4}$$

avec $K_i = \frac{M_I}{2^{b_i}}$, $K_j = \frac{M_J}{2^{b_j}}$, $\alpha = K_i k + m$ et $\beta = K_j l + n$.

Tout comme dans [Vai93], nous pourrions définir une procédure d'optimisation basée sur l'algorithme de Lloyd-max afin de calculer la partition centrale ainsi que les valeurs de reconstruction $\hat{x}_{k,l}$ qui minimisent la distorsion moyenne $\overline{d_{b_i b_j}}$ pour différentes combinaisons des nombres de plans de bits b_i et b_j. Cependant, le nombre important de combinaisons des valeurs b_i et b_j rend une telle optimisation complexe. De plus, les quantificateurs optimaux nécessitent la transmission des dictionnaires, ce qui dans ce cas représente une quantité importante d'information. Nous préférons donc adopter la solution la plus couramment utilisée en quantification scalaire à simple description, c'est à dire l'utilisation d'une quantification uniforme suivie d'un codage entropique. La partition centrale est donc entièrement définie par le pas de quantification Δ. Les valeurs de reconstruction sont calculées pour une source uniforme à partir des mesures de distorsions enoncées ci-dessus. Nous appelerons par la suite *MDUSQ* la quantification scalaire uniforme à descriptions multiples. Afin d'illustrer la validité de cette approche, la figure 2.5 présente un exemple de performances comparées entre la MDSQ et la MDUSQ pour une source gaussienne distribuée identiquement et indépendamment.

Nous essayons à présent d'évaluer les performances de la transmission progressive de MDUSQ à partir d'un exemple simple. Soit une source uniforme distribuée identiquement et indépendamment, quantifiée avec un pas de quantification Δ. Si on considère l'erreur quadratique moyenne (EQM) comme mesure de distorsion, alors, la distorsion moyenne dans le cas du codage à simple description est donnée par $\frac{\Delta^2}{12}$. La réception d'un plan de bit correspond à une division par deux du pas de quantification *delta*. On s'attend donc à voir la distorsion divisée par 4 à chaque plan de bit reçu. Le tableau 2.1 présente les distorsions calculées à l'aide de l'équation 2.4 en fonction de $\frac{\Delta^2}{12}$ avec Δ le pas de quantification le plus fin, pour une MDUSQ avec une table d'index dite "linéaire modifiée" [Vai93], $d = 2$ et $M_I = M_J = 8$ et pour toutes les valeurs possibles de b_i et b_j. On constate dans ce tableau que le gain apporté par la réception successive des plans de bits est irrégulier. En effet, les bits de poids fort apportent peu d'information. Il faut attendre les bits de poids faible pour obtenir une réduction importante de la distorsion. Par exemple, la réception simultanée des deux premiers plans de bits de chacune des descriptions permet de passer d'une distorsion de $1089\frac{\Delta^2}{12}$ à $183.86\frac{\Delta^2}{12}$, soit une réduction d'un facteur 5.9 de la distorsion. Par contre, la réception simultanée des deux derniers plans de bits de chacune des descriptions permet de passer d'une distorsion de $45.42\frac{\Delta^2}{12}$ à $\frac{\Delta^2}{12}$, soit une réduction d'un facteur 45.42. La réception progressive dans le cas normalement favorable du décodage central est donc peu efficace. On observe également de faibles performances pour d'autres scénarios de décodage. On retiendra que les bits de poids faibles "contiennent" plus d'information que les bits de poids fort.

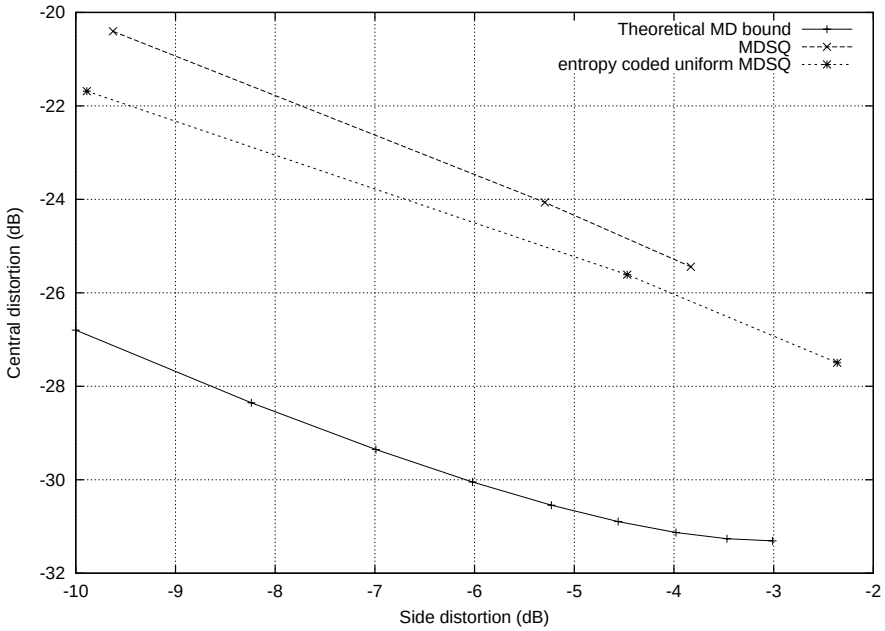


FIG. 2.5 – *Comparaison des rapports distorsions centrale/latérale de la MDSQ avec la MDUSQ (2.6 bit/échantillon/description).*

$b_j \backslash b_i$	3	2	1	0
3	$\frac{\Delta^2}{12}$	$16.09\ \frac{\Delta^2}{12}$	$65.12\ \frac{\Delta^2}{12}$	$111.19\ \frac{\Delta^2}{12}$
2	$16.09\ \frac{\Delta^2}{12}$	$45.42\ \frac{\Delta^2}{12}$	$104.79\ \frac{\Delta^2}{12}$	$163.26\ \frac{\Delta^2}{12}$
1	$65.12\ \frac{\Delta^2}{12}$	$104.79\ \frac{\Delta^2}{12}$	$183.86\ \frac{\Delta^2}{12}$	$320.29\ \frac{\Delta^2}{12}$
0	$111.19\ \frac{\Delta^2}{12}$	$163.27\ \frac{\Delta^2}{12}$	$320.29\ \frac{\Delta^2}{12}$	$1089\ \frac{\Delta^2}{12}$

TAB. 2.1 – *Distorsions en fonction de* $\frac{\Delta^2}{12}$ *pour une source uniforme quantifiée par MDUSQ avec table d'index linéaire modifiée,* $d = 2$ *et* $M_I = M_J = 8$.

Nous obtenons un résultat similaire par simulation pour une source gaussienne distribuée identiquement et indépendamment, de variance unité et de moyenne nulle (tableau 2.2). Pour cette source gaussienne, nous isolons dans le tableau 2.3 les résultats

$b_j \backslash b_i$	3	2	1	0
3	0.0033	0.0426	0.1095	0.2682
2	0.0427	0.1115	0.1856	0.3491
1	0.1093	0.1857	0.2570	0.5086
0	0.2683	0.3496	0.5091	1.0027

TAB. 2.2 – *EQM pour une source gaussienne quantifiée par MDUSQ avec table d'index linéaire modifiée,* $d = 2$ *et* $M_I = M_J = 8$.

des décodages central et latéraux pour les comparer à la limite théorique d'Ozarow [Oza80]. La encore, nous observons un écart important entre les performances obtenues

b_i, b_j	0	1	2	3
Décodage central	1.0027	0.2570	0.1115	0.0033
limite théorique	1	0.0626	0.0043	0.00031
Décodage latéral	1.0027	0.5091	0.3496	0.2683
limite théorique	1	0.25	0.0625	0.0156

TAB. 2.3 – *EQM pour une source gaussienne quantifiée par MDUSQ avec table d'index linéaire modifiée,* $d = 2$ *et* $M_I = M_J = 8$*, comparée à la limite théorique d'Ozarow [Oza80].*

et attendues. En particulier dans le cas du décodage central, il faut attendre la réception des deux bits de poids faibles pour s'approcher de la limite théorique. Dans le cas du décodage latéral, le résultat est en deçà de la limite théorique quel que soit le nombre de plans de bits reçus. Contrairement au décodage central, ce sont les bits de poids fort qui apportent le plus d'information.

Globalement, les résultats obtenus en codage progressif de MDUSQ sont peu satisfaisant. Dans la section suivante, nous expliquons ces résultats et proposons une solution adaptée.

2.2.3.2 MDUSQ progressive et tables d'index emboîtés

L'explication des contre-performances observées repose essentiellement sur la nature de la table d'index, qui constitue le point clé de la MDUSQ. Tout d'abord, la forme de la zone utile de la table d'index induit une forte redondance entre les bits de poids fort. En effet, comme cela est illustré par la figure 2.6, le fait d'occuper les diagonales qui entourent la diagonale principale implique que la majorité des index se trouvent réunis dans deux cadrans. De même, à l'intérieur de ces cadrans, la majorités des index sont réunis dans deux "sous-cadrans". Supposons par exemple, dans le cas de la figure 2.6, que les deux premiers bits de la description 1 d'un symbole sont reçus et sont égaux à 0.

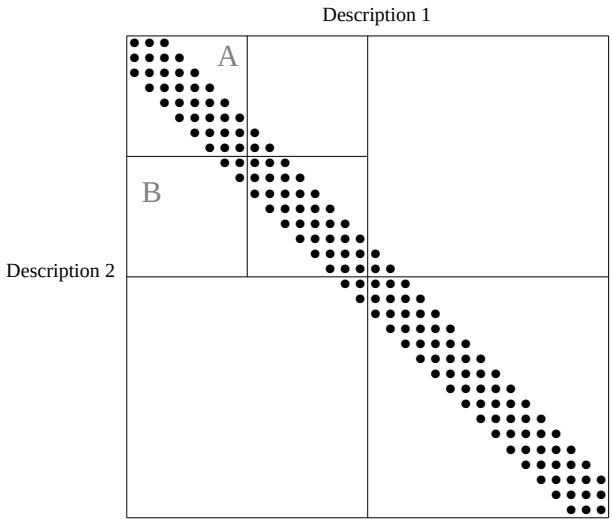


FIG. 2.6 – *Forme générale d'une table d'index MDSQ.*

On en déduit que l'intervalle codé se trouve dans le cadran A ou B. Par conséquent, le premier bit de la description 2 est forcément égal à 0. La réception de ce bit n'apporte strictement aucune information. De même, la réception du second bit de la description 2 n'apporte que peu d'information, puisque l'intervalle codé a une très forte probabilité d'être dans le cadran A. Il y a donc une très forte redondance entre les bits de poids fort des deux descriptions. Cette redondance diminue, lorsque la redondance globale de la table d'index diminue, c'est à dire lorsque l'on augmente d. Plus d est grand, plus il y a de cellules occupées dans la table d'index et moins il y a de redondance entre les bits de poids fort. De même, lorsque l'on reçoit les bits de poids faibles et si d n'est pas trop petit, il y a une forte probabilité pour que le cadran dans lequel se trouve l'intervalle codé soit presque totalement occupé. Il y a par conséquent peu, voire pas de redondance entre les bits de poids faible.

En résumé, en codage MDSQ, la redondance introduite entre les deux descriptions par la table d'index est répartie inégalement sur les bits des mots de codes. Les bits de poids forts sont plus redondants que les bits de poids faible. Cette propriété peut être considérée comme une qualité. En effet, en codage progressif, les bits de poids fort sont nécessaires pour exploiter les bits de poids faibles et sont donc plus importants. La MDSQ agit naturellement comme un schéma de protection inégale (section 1.3.2.1) et permet ainsi de garantir une qualité minimale au décodage.

Une deuxième explication aux contre-performances observées en codage progressif de MDUSQ repose sur la dispersion des intervalles dans la table d'index. En effet, dans la section 1.3.2.1, la dispersion a été définie comme la différence entre l'index maximal et l'index minimal des intervalles contenus dans une ligne ou une colonne de la table d'index. Les tables d'index sont conçues de manière à minimiser la dispersion maximale. Or, dans le cas de la transmission progressive, la dispersion est présente non

seulement en ligne et en colonne, mais aussi dans toute zone rectangulaire pouvant être adressée par la réception de b_i et b_j bits sur chaque description. Ainsi, dans l'exemple de la figure 2.4-b, la zone indiquée en gris foncé est composée d'intervalles disjoints. Cette situation désavantage en particulier le cas du décodage central par rapport à la MDUSQ non progressive. En effet, en MDUSQ non progressive, la partition centrale est toujours optimale. Ici, l'optimalité de la partition centrale n'est garantie que lorsque tous les plans de bits des deux descriptions ont été reçus.

A partir de cette seconde observation, nous proposons de concevoir une nouvelle table d'index adaptée à la transmission progressive de MDUSQ. Nous conserverons la forme diagonale de la table d'index, qui a le double avantage de permettre un contrôle aisé de la redondance et une protection plus importante des bits de poids fort. Notre effort se porte donc sur la minimisation de la dispersion. En particulier, nous souhaiterions éviter la présence d'intervalles de quantification disjoints dans le cas du décodage central, censé être le cas de décodage le plus favorable car sans perte de données. Nous proposons un algorithme de création de table d'index pour codage progressif qui respecte cette contrainte. Dans le cas du décodage latéral, nous comptons sur la forme diagonale de la table d'index pour éviter une dispersion trop importante. Pour les autres cas de décodage, nous remarquons qu'ils peuvent se décomposer successivement en décodage central, puis latéral : supposons par exemple que $b_i > b_j$. On a un décodage central pour les b_j premiers plans de bits, et un décodage latéral pour les $b_i - b_j$ suivants. L'algorithme proposé est donné dans le tableau 2.4. Il s'agit d'une fonction récursive F qui prend comme paramètres T la table d'index de taille $M_I \times M_J$, d la largeur de la diagonale couverte par les index et O_I, O_J les coordonnées de la paire d'index (i, j) située en haut et à gauche de T. Le paramètre *current* représente le prochain index à insérer dans T et est initialisé à 1. Du fait de sa construction récursive et de son utilité pour créer des flux de données dits *emboîtés*, nous appelons cette nouvelle stratégie de création de tables d'index *indexation emboîtée*. Un exemple de table d'index emboîtés est donné par la figure 2.7-b. La figure 2.7-a présente un exemple de table d'index classique issue de [Vai93] pour comparaison. Les tests effectués dans la section précédente permettent de vérifier l'intérêt de la stratégie d'indexation emboîtée. Tout d'abord, Le tableau 2.5 présente les distorsions calculées à l'aide de l'équation 2.4 en fonction de $\frac{\Delta^2}{12}$ avec Δ le pas de quantification le plus fin, pour une MDUSQ avec une table d'index emboîtés, $d = 2$ et $M_I = M_J = 8$ et pour toutes les valeurs possibles de b_i et b_j. La comparaison avec le tableau 2.1 montre l'apport de la table d'index emboîté. Du fait de la forme diagonale de l'index, le gain est limité pour les premiers plans de bits (réduction de la distorsion d'un facteur 7.09 lors de la réception des premiers bits de chaque description, contre 5.92 précédemment). Il est en revanche significatif pour les plans de bits intermédiaire. En effet, le passage de 1 à 2 bits reçus dans chaque description permet de réduire la distorsion d'un facteur 13.26 contre 4.05 précédemment.

La encore, un résultat similaire est obtenu par simulation pour une source gaussienne distribuée identiquement et indépendamment, de variance unité et de moyenne nulle (tableau 2.6). On remarque cependant que la table d'index emboîtés introduit un déséquilibre entre les deux descriptions. Pour cette source gaussienne, nous isolons dans le tableau 2.7 les résultats des décodages central et latéraux pour les comparer

```
F(T, d, O_I, O_J, M_I, M_J, current)
{
  if (M_I = 1) or (M_J = 1)
    for j from O_J to O_J + M_J − 1
    {
      for i from O_I to O_I + M_I − 1
      {
        if (j ≥ i − d) and (j ≤ i + d)
          T(i, j) := current
          current := current + 1
        endif
      }
    }
  else
    current := F(T, d, O_I, O_J, M_I/2, M_J/2, current)
    current := F(T, d, O_I + M_I/2, O_J, M_I/2, M_J/2, current)
    current := F(T, d, O_I, O_J + M_J/2, M_I/2, M_J/2, current)
    current := F(T, d, O_I + M_I/2, O_J + M_J/2, M_I/2, M_J/2, current)
— Remarque: alterner l'ordre de balayage à chaque niveau
de récursion permet de réduire la dispersion —
  endif
  return current
}
```

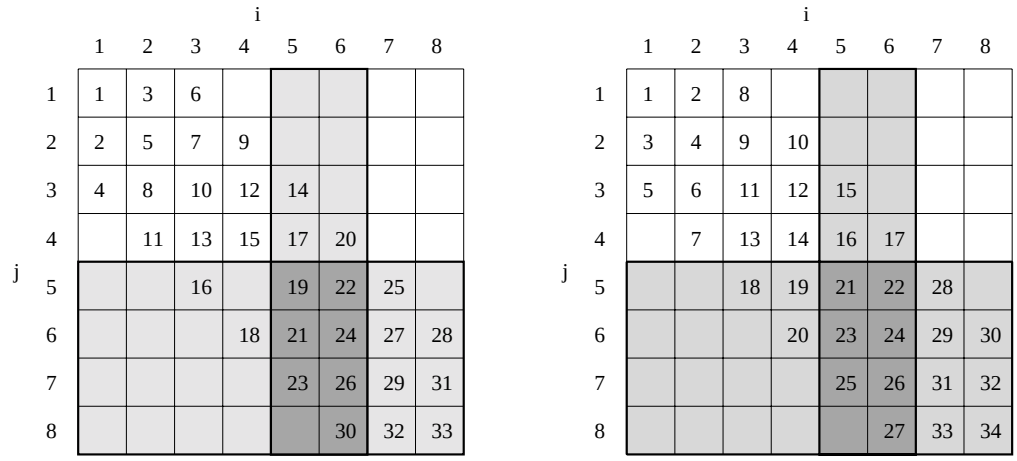
TAB. 2.4 – *Algorithme de création de tables d'index emboîtés.*

$b_j \backslash b_i$	3	2	1	0
3	$\frac{\Delta^2}{12}$	$7.71\ \frac{\Delta^2}{12}$	$33.47\ \frac{\Delta^2}{12}$	$115.04\ \frac{\Delta^2}{12}$
2	$3.47\ \frac{\Delta^2}{12}$	$12.29\ \frac{\Delta^2}{12}$	$41.94\ \frac{\Delta^2}{12}$	$136.81\ \frac{\Delta^2}{12}$
1	$94.06\ \frac{\Delta^2}{12}$	$114.55\ \frac{\Delta^2}{12}$	$163\ \frac{\Delta^2}{12}$	$393.64\ \frac{\Delta^2}{12}$
0	$140.53\ \frac{\Delta^2}{12}$	$173.11\ \frac{\Delta^2}{12}$	$289\ \frac{\Delta^2}{12}$	$1156\ \frac{\Delta^2}{12}$

TAB. 2.5 – *Distorsions en fonction de* $\frac{\Delta^2}{12}$ *pour une source uniforme quantifiée par MDUSQ avec table d'index emboîtés,* $d = 2$ *et* $M_I = M_J = 8$.

$b_j \backslash b_i$	3	2	1	0
3	0.0031	0.0097	0.0928	0.1928
2	0.0188	0.0313	0.1206	0.2338
1	0.0599	0.0771	0.1670	0.3645
0	0.3990	0.4450	0.6551	1.0027

TAB. 2.6 – *EQM pour une source gaussienne quantifiée par MDUSQ avec table d'index emboîtés,* $d = 2$ *et* $M_I = M_J = 8$.



(a) Table d'index linéaire modifiée [Vai93].

(b) Table d'index emboîtés.

FIG. 2.7 – *Tables d'index MDSQ pour codage progressif.*

à la limite théorique d'Ozarow [Oza80]. On observe un gain significatif dans le cas

b_i, b_j	0	1	2	3
Décodage central	1.0027	0.1670	0.0313	0.0031
limite théorique	1	0.0626	0.0043	0.00031
Décodage latéral	1.0027	0.5098	0.3394	0.2959
limite théorique	1	0.25	0.0625	0.0156

TAB. 2.7 – *EQM pour une source gaussienne quantifiée par MDUSQ avec table d'index emboîtés, $d = 2$ et $M_I = M_J = 8$, comparée à la limite théorique d'Ozarow [Oza80].*

du décodage central par rapport au résultats du tableau 2.3, pour une performance similaire dans le cas du décodage latéral.

Nous avons présenté le codage progressif par descriptions multiples, et montré qu'il était nécéssaire dans le cas de la MDUSQ d'adapter les tables d'index pour obtenir des résultats satisfaisants. Nous nous intéressons à présent à l'intégration du codage par descriptions multiples dans le codeur JPEG2000.

2.3 Enrichissement de JPEG2000 par codage à descriptions multiples

Avant de créer un codeur d'image à descriptions multiples, il est naturel de se demander si un tel codeur peut atteindre des performances intéressantes en compression.

Cette question à déjà été traitée dans [SRVN00], où il est mentionné que même si les bornes théoriques sur le codage par descriptions multiples ne sont valides qu'à haut débit, le comportement observé dans la pratique à bas débit est satisfaisant. Dans cette section, nous présentons deux codeurs d'image fixe à deux descriptions basés sur JPEG2000. En plus de performances débit/distorsion comparables à des codeurs similaires existants, ces deux codeurs héritent de toutes les fonctionnalités de JPEG2000, dont en particulier la progressivité.

2.3.1 Transformée polyphase et quantification sélective

L'intégration du codage à deux description par transformée polyphase et quantification sélective dans un codeur JPEG2000 est résumée par la figure 2.8. Les modifications

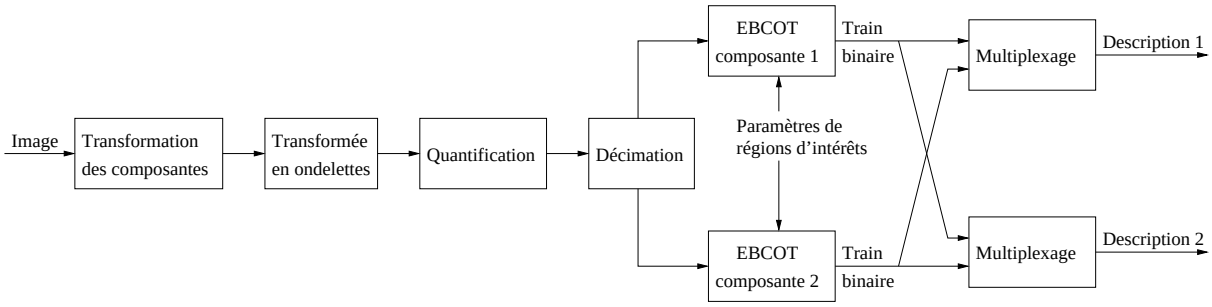


FIG. 2.8 – *Schéma bloc du codeur à descriptions multiples par transformée polyphase basé JPEG2000*

apportées au codeur JPEG2000 sont mineures. En effet, seule la décimation intervient après la quantification. Cette décimation par deux est effectuée en ligne, ce qui revient à placer les colonnes de chaque sous-bande alternativement dans les deux composantes polyphases. Ces deux composantes sont codées indépendamment par l'algorithme EBCOT. L'optimisation débit/distorsion d'EBCOT peut être effectuée indépendamment car les deux composantes polyphases ne contiennent pas les mêmes échantillons et la mesure de distorsion utilisée par EBCOT est supposée additive. Minimiser la distorsion induite par les deux composantes polyphases revient donc à minimiser la distorsion pour chacune de ces composantes.

Comme cela a été expliqué en section 2.2.2, la redondance entre les deux descriptions est contrôlée en combinant le principe de la quantification sélective et la propriété de progressivité des flux binaires générés par EBCOT. L'entrelacement des deux composantes polyphases est effectué a posteriori, indépendamment d'EBCOT. Ce choix permet d'avoir un système particulièrement flexible, puisque l'image n'a pas besoin d'être re-compressée pour modifier le taux de redondance.

Enfin on peut signaler pour conclure le fait que la transformée polyphase est appliquée dans le domaine ondelette. Il est également possible de l'appliquer dans le domaine spatial. Cependant, cela suppose une approche légèrement différente. En effet, l'importante corrélation présente dans les images implique une forte redondance entre

les deux composantes polyphases. Comme il n'est pas possible de s'affranchir de cette redondance, l'intérêt d'un tel schéma est limité. Si cependant il devait être utilisé, alors la redondance entre les deux composantes polyphases pourrait être exploitée par des techniques d'estimation ou d'interpolation. Cette redondance pourrait être augmentée en codant la différence entre les composantes polyphases sources et interpolées. De tels schémas ont été testés durant la thèse et les résultats sont détaillés dans [CTK01]. Le système présenté ici est le plus intéressant et le seul qui a été retenu.

2.3.2 MDUSQ

Le codeur MDUSQ basé JPEG2000 proposé est résumé par la figure 2.9. Cette

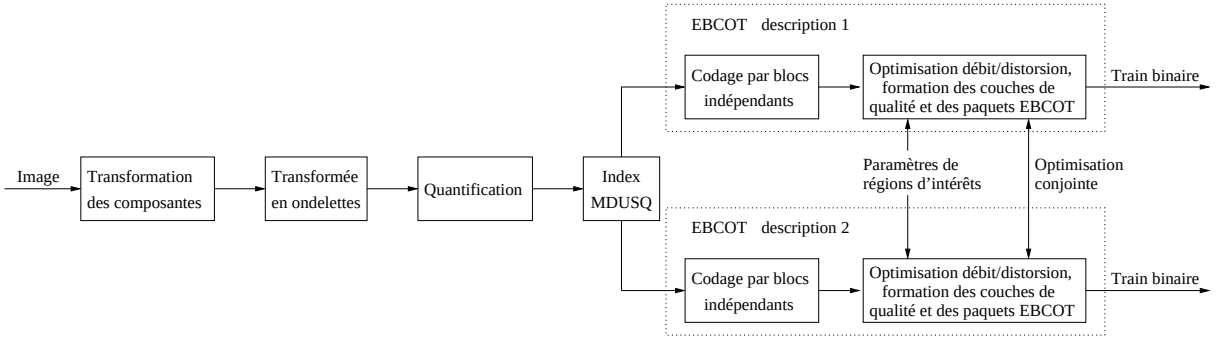


FIG. 2.9 – *Schéma bloc du codeur MDUSQ basé JPEG2000*

structure découle naturellement de l'utilisation de la MDUSQ. Afin d'accroître la robustesse du système, nous avons appliqué la MDUSQ uniquement sur l'amplitude des coefficients quantifiés. Le signe de ces coefficients est dupliqué dans les deux descriptions. La taille de la table d'index est calculée en fonction de la plus grande amplitude à coder. La redondance est contrôlée par l'intermédiaire du paramètre d représentant le nombre de diagonales occupées dans l'index MDUSQ. Lors du décodage, l'algorithme EBCOT fournit le nombre de bits reçus pour chaque échantillon de chaque description. Les valeurs de reconstruction sont calculées à partir de cette information et de l'équation 2.4, en supposant la source uniforme.

Si les modifications apportées au codeur peuvent paraître minimes, l'ajout de la MDUSQ peut avoir un impact important sur les performances du codeur.

2.3.2.1 Influence de la MDSQ sur la statistique de la source

L'utilisation par EBCOT du codage arithmétique nécessite de disposer d'un modèle de source. Un modèle binaire contextuel adapté au codage par plans de bits en trois passes des code-block est fixé dans le standard JPEG2000. C'est un modèle général, fixé a priori pour tout type d'image et indépendant du niveau de résolution de la transformée en ondelettes et du plan de bits. La question est de savoir si ce modèle, qui a été conçu pour coder une source issue d'une quantification scalaire, est aussi

adapté au codage d'une description générée par MDUSQ. Nous apportons un élément de réponse en comparant les distributions issues d'une quantification scalaire et d'une MDUSQ. Plus précisément, nous cherchons à mesurer si la distribution d'une source quantifiée par MDUSQ est éloignée de celle d'une source qui a subi une quantification scalaire classique.

	$d=2$	$d=4$	$d=8$
Description 1	0.003361	0.01721	0.02706
Description 2	0.006858	0.02008	0.14882

TAB. 2.8 – *Entropie relative entre les distributions issues des quantifications scalaire et MDUSQ ($M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$ (en bits par échantillons).*

Considérons une source laplacienne S de paramètre λ distribuée identiquement et indépendamment. Soit un index MDUSQ A avec $2d+1$ diagonales couvertes et $M_I = M_J$. Soit P la distribution stationnaire des symboles issus de la quantification scalaire uniforme de S sur M_I intervalles. Soient P_1 et P_2 les distributions stationnaires des symboles des deux descriptions obtenues à partir de A. La distance de Kullback-Leibler ou entropie relative permet de mesurer la divergence entre deux distributions. Le tableau 2.8 donne les entropies relatives $D(P||P_1)$ et $D(P||P_2)$ obtenues pour $\lambda = 1$, $M_I = M_J = 32$ et différentes valeurs de d. Les distributions correspondant à ces diférentes valeurs de d sont tracées sur les figures 2.10, 2.11 et 2.12. Les mêmes observations ont été faites pour d'autres valeurs de M_I, M_J, d et λ. Nous remarquons principalement que moins il y a de redondance entre les deux descriptions, plus les distributions sont éloignées. Cela s'explique naturellement par la dispersion de la table d'index MDUSQ qui augmente avec d. Dans le cadre de l'intégration de la MDUSQ dans JPEG2000, on s'attend donc à voir des performances en compression non dégradées, à condition d'avoir suffisament de redondance entre les deux descriptions. Le codage de deux descriptions non corrélées nécessite une adaptation du modèle de source binaire contextuel.

2.3.2.2 Optimisation débit distorsion

L'optimisation débit/distorsion n'est pas spécifiée dans le standard JPEG2000, mais son utilisation est implicite. C'est ce qui permet à l'algorithme EBCOT de générer des trains binaires progressifs optimaux. Nous reprenons ici l'optimisation débit/distorsion proposée dans [Tau00] et utilisée dans le "verification model" (VM) de JPEG2000, pour ensuite l'étendre au cas du codage MDUSQ. L'algorithme EBCOT découpe les sous-bandes de la transformée en ondelettes de l'image en code-blocks B_i. Pour chacun de ces code-blocks, un train binaire progressif qui peut être tronqué à différents débits R_i^n est généré. La contribution d'un code-block B_i à la distorsion globale D de l'image reconstruite est notée D_i^n, pour chaque point de coupure n. L'ensemble des paires débit/distorsion $\{R_i^n, D_i^n\}$ est calculé au moment du codage de B_i et conservé. La

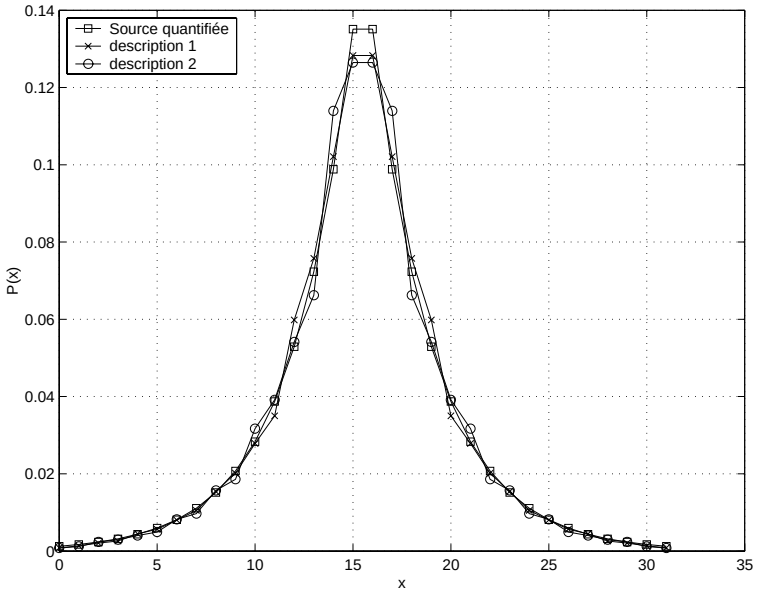


FIG. 2.10 – *Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 2$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$.*

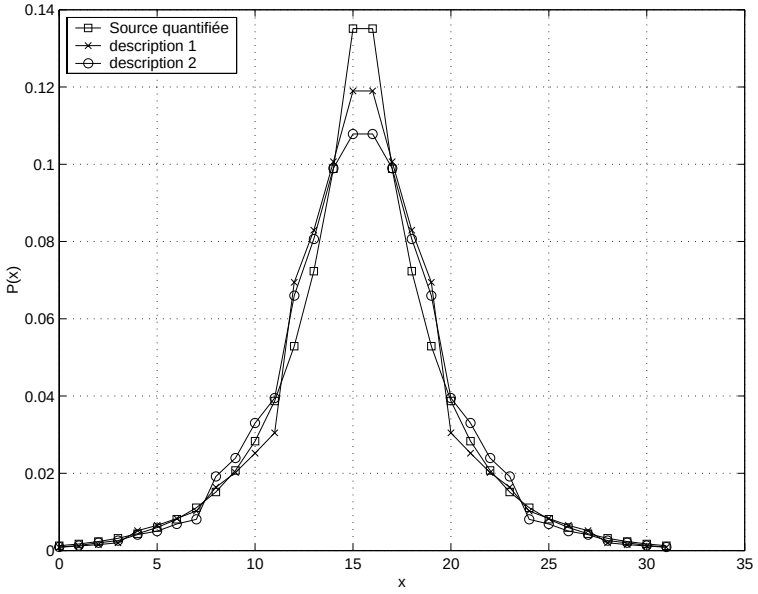


FIG. 2.11 – *Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 4$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$.*

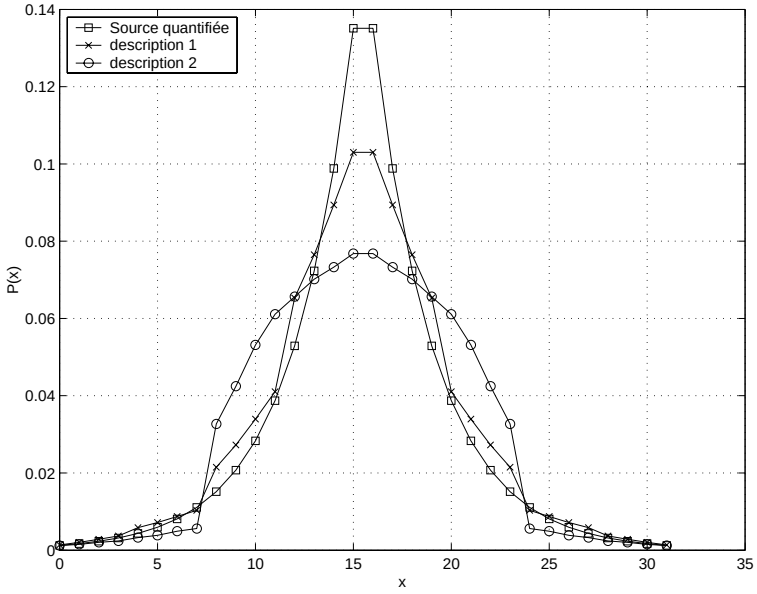


FIG. 2.12 – *Comparaison des distributions issues des quantifications scalaire et MDUSQ (diagonale $d = 8$, $M_I = M_J = 32$) d'une distribution laplacienne de paramètre $\lambda = 1$.*

métrique de distorsion utilisée est supposée additive:

$$D = \sum_i D_i^{n_i}, \tag{2.5}$$

avec n_i le point de coupure choisi pour le code-block B_i. La métrique de distorsion que nous utilisons est l'erreur quadratique moyenne (EQM). La contribution de B_i à la distorsion globale est donnée par

$$D_i^n = w_{b_i}^2 \sum_{k \in B_i} (\hat{s}_i^n[k] - s_i[k])^2, \tag{2.6}$$

avec k l'index des symboles dans le code-block B_i et $\hat{s}_i^n[k]$ la valeur reconstruite correspondant au symbole $s_i[k]$ quantifié et tronqué au point de coupure n. Le coefficient $w_{b_i}^2$ représente la norme L2 de la base de fonctions ondelettes pour la sous-bande b_i à laquelle appartient B_i.

Le problème d'optimisation posé consiste à trouver l'ensemble des points de coupures n_i qui minimise la distorsion globale D sous la contrainte d'un débit maximal $R \leq R^{max}$. La méthode des multiplicateurs de Lagrange est utilisée. Un ensemble de points de coupure $\{n_i^\lambda\}$ qui minimise

$$D(\lambda) + \lambda R(\lambda) = \sum_i (D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda}) \tag{2.7}$$

est optimal dans le sens où la distorsion ne peut pas être réduite sans augmenter le débit et vice-versa. La solution finale est obtenue en trouvant la plus petite valeur de λ pour laquelle $R(\lambda) \leq R^{max}$. Les détails de cet algorithme sont donnés dans [Tau00] et [Eve63].

Dans le cas du codage MDUSQ, nous avons deux descriptions $B_{1,i}$ et $B_{2,i}$ de chaque code-block, pouvant être tronquées respectivement aux débits $R_{1,i}^{n_1}$ et $R_{2,i}^{n_2}$. Soient les points de coupures n_1 et n_2 associés respectivement aux code-blocks $B_{1,i}$ et $B_{2,i}$. On peut calculer la contribution de ces deux code-blocks à la distorsion globale D soit dans le cas général ($D_{0,i}^{n_1,n_2}$) soit dans le cas latéral ($D_{1,i}^{n_1}$ et $D_{2,i}^{n_2}$). Ces trois distorsions sont données par

$$\begin{aligned} D_{0,i}^{n_1,n_2} &= w_{b_i}^2 \sum_{k \in B_i} (\hat{s}_{0,i}^{n_1,n_2}[k] - s_i[k])^2, \\ D_{1,i}^{n_1} &= w_{b_i}^2 \sum_{k \in B_i} (\hat{s}_{1,i}^{n_1}[k] - s_i[k])^2, \\ D_{2,i}^{n_2} &= w_{b_i}^2 \sum_{k \in B_i} (\hat{s}_{2,i}^{n_2}[k] - s_i[k])^2. \end{aligned} \tag{2.8}$$

De la même manière, on peut calculer la distorsion globale soit dans le cas général, soit dans le cas latéral :

$$\begin{aligned} D_0 &= \sum_i D_{0,i}^{n_{1,i},n_{2,i}}, \\ D_1 &= \sum_i D_{1,i}^{n_{1,i}}, \\ D_2 &= \sum_i D_{2,i}^{n_{2,i}}. \end{aligned} \tag{2.9}$$

Notons que le cas du décodage central est un cas particulier où les débits globaux des deux descriptions sont égaux : $R_1 = R_2$. Cela n'implique pas que les débits et les points de coupures par blocs soient égaux : $n_{1,i} \neq n_{2,i}$ et $R_{1,i}^{n_{1,i}} \neq R_{2,i}^{n_{2,i}}$.

Le problème d'optimisation dans le cas de la MDUSQ peut consister à trouver l'ensemble des points de coupures $\{n_{1,i}, n_{2,i}\}$ qui minimise D_0 pour une contrainte de débit donnée $R_1 \leq R_1^{max}$ et $R_2 \leq R_2^{max}$. L'optimisation débit/distorsion est valide dans la mesure où l'on retrouve au décodage la situation dans laquelle cette optimisation a été effectuée au codage. Dans le cas du codage par descriptions multiples, on souhaite décoder l'image sans connaissance a priori de la contribution apportée par chaque description. On ne peut donc pas prévoir au moment du codage les paires de débits (R_1^{max}, R_2^{max}) pour lesquelles il faut minimiser la distorsion. Pour résoudre ce problème, nous supposons que si les trains binaires des deux descriptions ont été optimisés pour un contexte de décodage donné, cela ne dégradera pas ou peu les performances dans les autres contextes de décodage. Les résultats ainsi obtenus sont satisfaisants. Nous proposons trois stratégies d'optimisation débit/distorsion pour le codeur d'image basé MDUSQ, toutes basées sur l'utilisation de la méthode des multiplicateurs de Lagrange :

1. l'optimisation basée sur le décodage central consiste à trouver un ensemble de points de coupure $\{n_{1,i}^{\lambda}, n_{2,i}^{\lambda}\}$ qui minimise $D_0(\lambda) + \lambda(R_1(\lambda) + R_2(\lambda))$ sous la contrainte $R_1 \leq R^{max}$ et $R_2 \leq R^{max}$.

2. L'optimisation basée à la fois sur les décodages central et latéraux consiste à trouver un ensemble de points de coupure $\{n_{1,i}^{\lambda}, n_{2,i}^{\lambda}\}$ qui minimise $D(\lambda) + \lambda(R_1(\lambda) + R_2(\lambda))$, avec $D(\lambda) = D_0(\lambda) + \phi(D_1(\lambda) + D_2(\lambda))$ et sous la contrainte $R_1 \leq R^{max}$ et $R_2 \leq R^{max}$. Cette formulation est inspirée par [Vai93].
3. Enfin, l'optimisation indépendante des deux descriptions consiste à considérer chaque description comme un train binaire unique issu d'un codage à simple description. Par exemple, pour la description 1, cela revient à trouver un ensemble de points de coupure $\{n_{1,i}^{\lambda}\}$ qui minimise $D_1(\lambda) + \lambda R_1(\lambda)$ sous la contrainte $R_1 \leq R^{max}$.

Nous avons utilisé cette dernière stratégie pour nos expériences. Elle a l'avantage de chercher deux ensembles de points de coupures distincts, plutôt qu'un ensemble unique de paires de points de coupure. Sa complexité est donc moindre (deux fois $O(N)$ au lieu de $O(N^2)$). Bien entendu, ce choix peut être remis en cause suivant le contexte d'utilisation du codeur d'image MDUSQ basé JPEG2000.

2.3.3 Allocation de redondance

La redondance entre les deux descriptions est ajustée par le paramètre d pour la MDUSQ. Pour la transformée polyphase, l'allocation optimale de redondance est calculée dans [JO99] en fonction de la probabilité p de perte d'une description. Pour la MDUSQ en revanche, l'allocation de redondance est calculée d'une manière légèrement différente dans [BV97]. A partir des résultats de [BV97], il est possible d'exprimer le paramètre d en fonction de p.

D'après [BV97], les distorsions centrales D_c et latérales D_s d'une MDSQ avec $M_I = M_J = 2^q$ sont données par

$$\begin{aligned} D_c &= C\sigma^2 2^{-2q(1+\frac{1}{n})}, \\ D_s &= S\sigma^2 2^{-2q(1-\frac{1}{n})}, \end{aligned} \tag{2.10}$$

avec $d = 2^{\frac{q}{n}}$, σ^2 la variance de la source et avec C et S déterminés par la fonction de densité de probabilité de la source et par la méthode de codage. De la même manière que dans [JO99], on cherche à minimiser la fonction de coût $J = D_c + \lambda D_s$ en posant $\frac{\delta J}{\delta n}|_{n=n^*} = 0$, ce qui conduit à

$$\frac{1}{n^*} = -\frac{1}{4q} log_2 \frac{\lambda S}{C}. \tag{2.11}$$

La probabilité de perte d'une description est introduite en calculant la distorsion moyenne

$$D = (1-p)^2 D_c + 2p(1-p)D_s + p^2\sigma^2, \tag{2.12}$$

dont la minimisation conduit à

$$\frac{1}{n^*} = -\frac{1}{4q} log_2 \frac{2pS}{(1-p)C}, \tag{2.13}$$

d'où

$$d^* = 2^{-\frac{1}{4} log_2 \frac{2pS}{(1-p)C}}. \tag{2.14}$$

On remarque que dans une application pratique, il est nécessaire de discrètiser le paramètre d^* ainsi obtenu. Ce calcul peut être généralisé au codage par transformation en cherchant une valeur optimale de d pour chaque sous-bande [BV97].

2.4 Résultats expérimentaux

Deux codeurs d'image fixe à deux descriptions basé JPEG2000 ont été mis en oeuvre. L'un utilise la MDUSQ et l'autre la transformée polyphase couplée à de la quantification sélective. Nous appelons ces deux codeurs respectivement JPEG2000-MDUSQ et JPEG2000-PT. Différentes expériences ont été réalisées pour valider ces deux codeurs. Nous présentons ici les résultats les plus significatifs obtenus en codant l'image Lena de taille 512×512 pixels. Sur chaque courbe, nous indiquons à titre indicatif la limite de performance imposée par le codeur JPEG2000. Les codeurs développés étant basés sur JPEG2000, ils ne peuvent pas donner de meilleur résultats en terme de rapport débit/distorsion que le codeur original. L'apport du codage par descriptions multiples se situe au niveau de la robustesse. L'ajout de redondance entre les deux descriptions implique une perte en efficacité de compression. La comparaison avec le codeur original permet d'évaluer cette perte.

Afin d'évaluer l'efficacité du codage progressif, nous présentons des courbes débit/distorsion pour un niveau de redondance fixé, dans les cas particuliers du décodage central et du décodage latéral. Les résultats dans le cas du décodage latéral sont donnés par rapport au *débit central* correspondant. Considérons par exemple que la description 1 est reçue avec un débit R. On suppose alors que la description 2 a également été transmise avec un débit R. Le débit total transmis est donc $2R$. Ainsi, quand on ne décode qu'une seule description, on n'exploite que la moitié du débit $2R$ qui a été transmis. La limite de performance imposée par le codeur JPEG2000 dans ce cas correspond au décodage de l'image avec le débit R. La limite de performance dans le cas du décodage latéral correspond à la redondance maximale, c'est à dire à la duplication de l'information.

Les figures 2.13 et 2.14 présentent les courbes débit/distorsions obtenues avec le codeur JPEG2000-MDUSQ respectivement dans le cas central et dans le cas latéral, avec les tables d'index linéaire modifiée [Vai93] et emboîtée, pour $d = 4$. Le gain apporté par la stratégie d'indexation emboîtée est évident aussi bien en décodage central qu'en décodage latéral. Le déséquilibre entre les deux descriptions induit par l'index emboîté est significatif, mais reste acceptable. Les résultats obtenus avec la table d'index linéaire modifiée montrent que celle-ci n'est absolument pas adaptée au décodage progressif. En effet, à bas débit, le décodage central est à peine meilleur que le cas limite de la duplication de l'information.

Les figures 2.15 et 2.16 permettent de comparer les performances des deux codeurs proposés pour une redondance équivalente entre les deux descriptions. La figure 2.15 montre des performances équivalentes dans le cas du décodage central. Dans le cas du

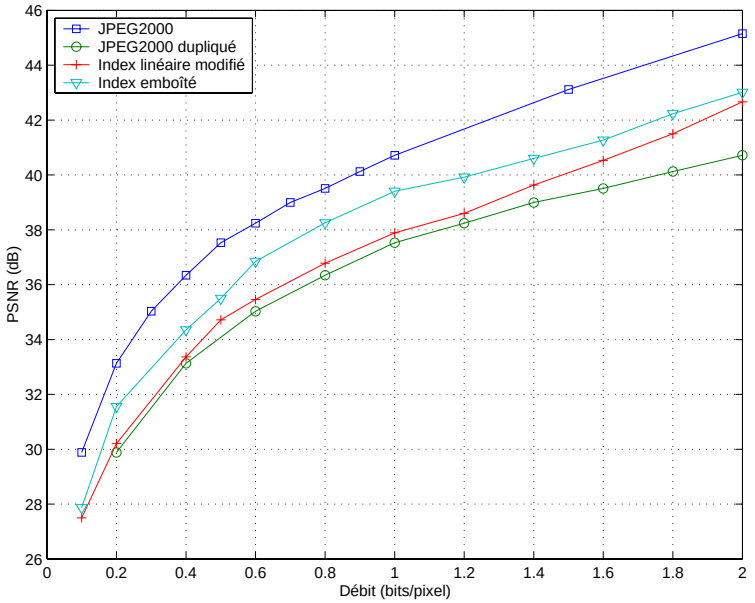


FIG. 2.13 – *Courbes débit/distorsion pour le décodage central de Lena 512, comparaison des tables d'index linéaires modifiées et emboîtées ($d = 4$).*

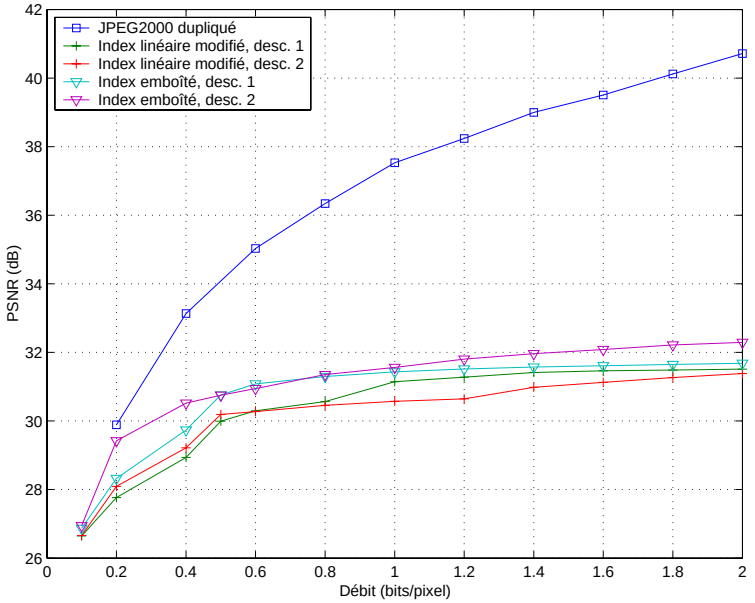


FIG. 2.14 – *Courbes débit/distorsion pour le décodage latéral de Lena 512, comparaison des tables d'index linéaires modifiées et emboîtées ($d = 4$).*

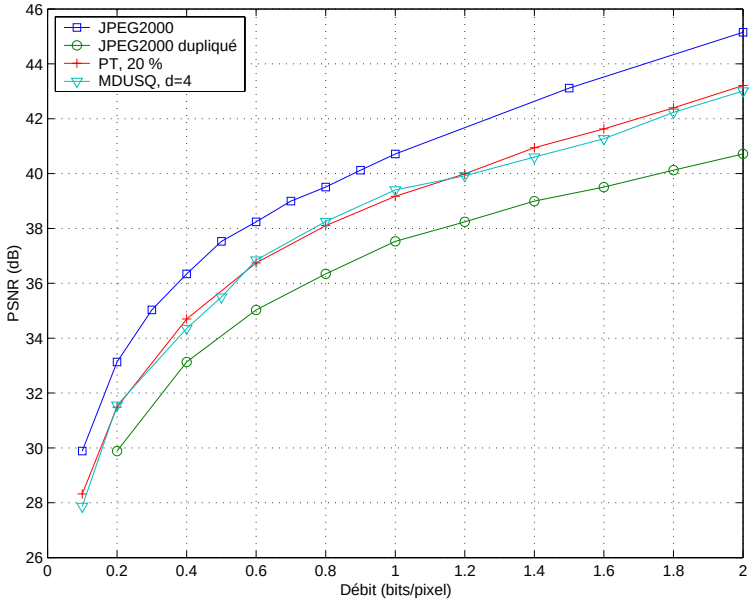


FIG. 2.15 – *Courbes débit/distorsion pour le décodage central de Lena 512, comparaison de la transformée polyphase et de la MDUSQ.*

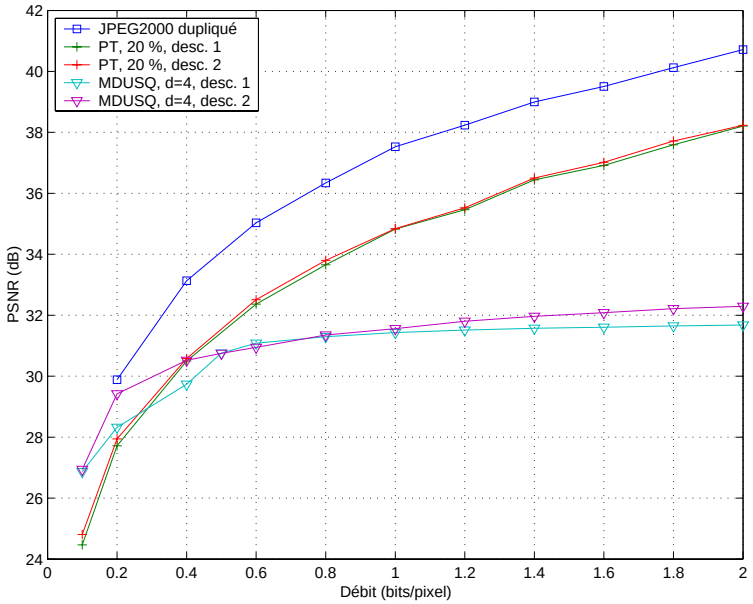


FIG. 2.16 – *Courbes débit/distorsion pour le décodage latéral de Lena 512, comparaison de la transformée polyphase et de la MDUSQ.*

décodage latéral, en revanche, la figure 2.16 montre que le codeur JPEG2000-MDUSQ est plus performant que le codeur JPEG2000-PT pour un débit inférieur ou égal à 0.3 bits/pixel (soit 0.15 bits/pixel/description) et moins performant sinon. Le comportement du codeur JPEG2000-MDUSQ est peu avantageux pour les hauts débits. Cette mauvaise performance est inhérente à la MDUSQ, qui impose des intervalles de quantification disjoints dans le cas du décodage latéral. Le même comportement est observable sur des simulations. Ainsi, dans le tableau 2.7, dans le cas du décodage latéral, la réception du troisième plan de bit n'amène que peu d'information. L'intérêt du codeur JPEG2000-MDUSQ est donc limité au codage d'image à bas débit.

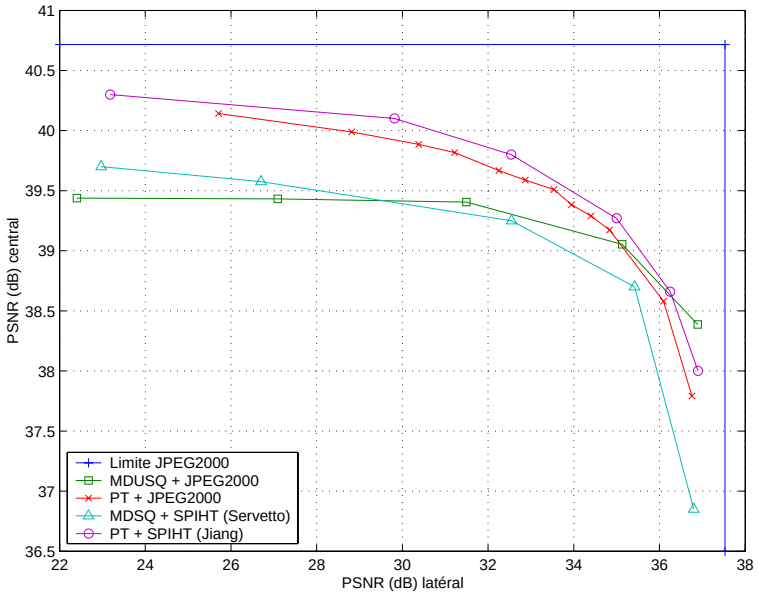


FIG. 2.17 – *Compromis entre les distorsions centrale et latérale en fonction de la redondance dans le cas du décodage de Lena 512 à 0.5 bits/pixel, comparaison entre les deux codeurs proposés et deux codeurs de référence ([SRVN00] et [JO99]).*

Le compromis entre les distorsions centrales et latérales permet d'évaluer les performances d'un système de codage par descriptions multiples en fonction de la redondance entre les descriptions. Les figures 2.17 et 2.18 présentent ce compromis respectivement à 0.5 et 0.25 bits par pixel par description, pour les deux codeurs proposés et pour deux autres codeurs de référence. Ces codeurs sont tous les deux basés sur des codeurs d'image fixe de type SPIHT. Le premier [SRVN00] utilise la MDSQ et le second [JO99] la transformée polyphase. Le cas de la réception progressive des deux descriptions n'est pas traité explicitement. Il est implicite dans le cas de la transformée polyphase. Notons que les limites indicatives données par le codeur JPEG2000 ne doivent pas être prises en compte pour les codeurs de type SPIHT. En effet, SPIHT se montre un peu plus performant que JPEG2000 en terme de rapport débit/distorsion. Cette différence se retrouve dans les figures 2.17 et 2.18, excepté pour les fortes redondance

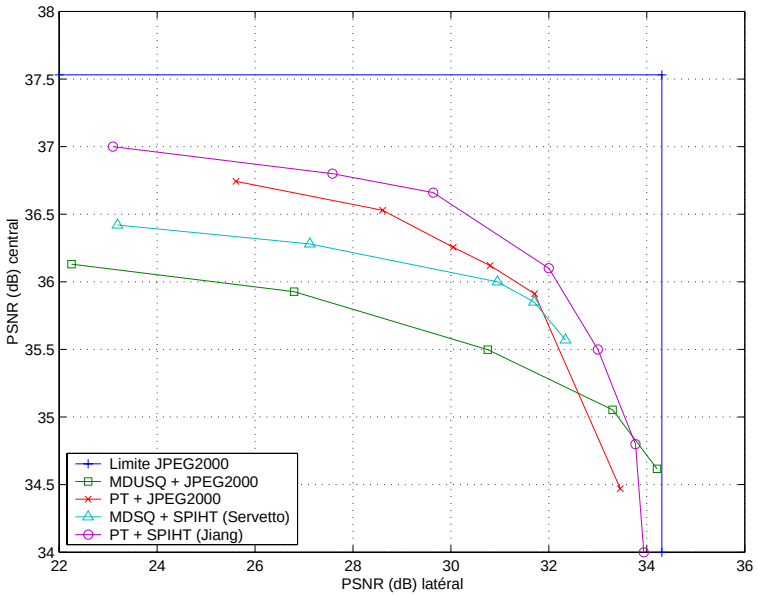


FIG. 2.18 – *Compromis entre les distorsions centrale et latérale en fonction de la redondance dans le cas du décodage de Lena 512 à 0.25 bits/pixel, comparaison entre les deux codeurs proposés et deux codeurs de référence ([SRVN00] et [JO99]).*

où le codeur JPEG2000-MDUSQ surpasse tous les autres. Nous avons prédit dans la section 2.3.2.1 de moins bonnes performances du codeur JPEG2000-MDUSQ pour les faibles redondances. Cette prédiction se retrouve expérimentalement. On peut donc supposer qu'il est possible d'améliorer les performances du codeur JPEG2000-MDUSQ pour les fortes redondances en adaptant les contextes du codeur arithmétique. Les deux codeurs proposés apportent un bon compromis entre les performances débit/distorsion et la robustesse tout en conservant toutes les caractéristiques de JPEG2000.

débit 1 \ débit 2	0.5	0.4	0.3	0.2	0.1	0
0.5	39.4053	38.4785	37.4473	36.4722	34.6979	31.4318
0.4	39.1445	38.2549	37.2258	36.2407	34.4888	31.2971
0.3	38.6767	37.8324	36.845	35.9158	34.1852	31.0812
0.2	36.3783	35.8483	35.0947	34.3531	33.3735	29.7368
0.1	33.7025	33.3758	32.885	32.3427	31.5552	28.3242
0	31.56	31.3533	30.9449	30.5192	29.423	14.7602

TAB. 2.9 – *PSNR en réception inégale de deux descriptions de Lena 512 obtenues avec le codeur JPEG2000-MDUSQ ($d = 4$).*

Les tableaux 2.9 et 2.10 illustrent la possibilité de réception inégale des deux descriptions, respectivement pour les codeurs JPEG2000-MDUSQ et JPEG2000-PT. Cette possibilité n'a pas été étudiée par les auteurs d'autres schémas de codage à descriptions

débit 1 \ débit 2	0.5	0.4	0.3	0.2	0.1	0
0.5	39.2288	38.6875	37.8381	36.6575	34.4686	33.5693
0.4	38.7315	38.2284	37.4553	36.3517	34.1878	32.1595
0.3	37.8524	37.4339	36.7643	35.7963	33.8284	31.6431
0.2	36.6946	36.3594	35.8174	34.982	33.2569	29.2091
0.1	34.4538	34.1326	33.79	33.2133	31.9369	26.3109
0	33.3359	32.1426	31.6327	29.1511	26.4048	14.7602

TAB. 2.10 – *PSNR en réception inégale de deux descriptions de Lena 512 obtenues avec le codeur JPEG2000-PT (20% de redondance).*

multiples, tels que [SRVN00] et [JO99]. On retrouve les résultats déjà observés dans les cas particuliers du décodage central et du décodage latéral. On remarque dans le cas général la supériorité du décodage central. Plus les descriptions sont équilibrées, meilleur est le résultat. Pour un débit total fixé $R = R_1 + R_2$, le résultat est d'autant meilleur que la différence $|R_1 - R_2|$ est faible, l'optimum étant atteint pour $R_1 = R_2$.

Enfin, pour terminer, les figures 2.19 et 2.20 illustrent visuellement le comportement des deux codeurs proposés. Les résultats observés sont comparables.

2.5 Conclusion

Dans ce chapitre, nous avons proposé l'intégration de deux techniques de codage par descriptions multiples dans un codeur d'image JPEG2000. Dans le cas de la quantification scalaire à descriptions multiples, nous avons proposé une nouvelle stratégie d'indexation qui permet le codage progressif. En plus de performances intéressantes en compression, les deux codeurs conservent toutes les fonctionnalités de JPEG2000. Ces deux codeurs sont naturellement robustes aux effacements, à condition de respecter l'hypothèse de l'erreur résiduelle nulle sur chaque description. Si les conditions de transmissions sont défavorables, comme par exemple sur des réseaux sans fil, le respect de cette contrainte impose l'ajout d'une redondance importante sous forme de codes de canal, en plus de la redondance introduite par le codage par descriptions multiples. Les performances en compression d'un tel système deviennent alors peu intéressantes.

Si on lève l'hypothèse de l'erreur résiduelle nulle, il faut alors introduire un autre mécanisme permettant la correction d'erreurs. La redondance introduite par le codage à descriptions multiples peut permettre de détecter des erreurs et éventuellement de les corriger ou de limiter leur impact. Par exemple, une table d'index MDUSQ peut servir de détecteur d'erreur. En effet, si pour un symbole s les deux index i et j sont reçus, et si l'un des deux index est erroné, alors deux situations sont possibles:

- la paire (i, j) adresse un emplacement inoccupé de la table d'index. Dans ce cas, on détecte une erreur.
- la paire (i, j) adresse un emplacement occupé de la table d'index. Alors l'erreur n'est pas détectée, mais son impact est limité de par la construction de la table d'index (minimisation de la dispersion).

(a) Original. (b) Décodage central.

(c) Décodage description 1. (d) Décodage description 2.

FIG. 2.19 – *Codage JPEG2000-MDUSQ de Lena 512 avec $d = 4$ et 0.25 bits par pixel par description.*

(a) Original. (b) Décodage central.

(c) Décodage description 1. (d) Décodage description 2.

FIG. 2.20 – *Codage JPEG2000-PT de Lena 512 avec 20% de redondance et 0.25 bits par pixel par description.*

Une telle solution n'est cependant pas efficace dans l'application considérée. En effet, comme cela est visible sur les figures 2.8 et 2.9, le codage par descriptions multiples se situe en amont du bloc EBCOT dans la chaîne de codage. Or, ce bloc EBCOT contient une étape de codage arithmétique qui est particulièrement sensible aux erreurs. Une seule erreur bit entraîne la désynchronisation du décodeur et se propage jusqu'à la fin de la séquence décodée, d'où une augmentation dramatique du nombre d'erreurs. On ne peut pas espérer gérer une telle quantité d'erreurs simplement en utilisant le codage par descriptions multiples. Il faut donc traiter spécifiquement la fragilité des codes arithmétiques face aux erreurs bits.

Chapitre 3

Décodage souple de codes arithmétiques

3.1 Introduction

Le codage entropique, ou codage à longueur variable est un composant incontournable dans une majorité de schémas de compression. Cependant, les codes à longueurs variables (VLC) sont particulièrement sensibles aux erreurs de transmission : un bit altéré par le canal suffit à provoquer une perte de synchronisation du décodeur. Les frontières des symboles ne sont plus localisées correctement, ce qui conduit à des taux d'erreur inacceptables. Un travail de recherche important a été effectué pour concevoir des codes qui ont de meilleures propriétés de synchronisation [LR92,TF84,YTM95, WV98], de même que des algorithmes de décodage souple [SV01,MP98,PM98] et de décodage conjoint source/canal [MF98,DS98,BH00b,GFGR01] de VLCs. Les recherches se sont d'abord focalisées sur les codes de Huffman [PM98,MF98,DS98,BH00a] et sur les VLCs réversibles [YTM95,WV98,BH00b].

Plus récemment, les codes arithmétiques ont reçu un regain de popularité dans des applications pratiques, incluant les standards JPEG2000 [ISO00], H264 et MPEG4 [Joi02]. Les codes arithmétiques ont la propriété d'atteindre des performances arbitrairement proches de l'entropie. Lorsqu'ils sont couplés à des modèles statistiques de sources d'ordres supérieur, des facteurs de compression considérables peuvent être atteints. La contrepartie de ces performances est une sensibilité accrue au bruit. La différence majeure entre le codage arithmétique et le codage de Huffman réside dans le fait qu'en codage arithmétique, il n'y a pas de correspondance un à un entre les symboles sources et les mots de codes binaires, mais une correspondance entre la totalité de chaque séquence possible et le train binaire associé à cette séquence. Cette contrainte impose une modélisation et un traitement différent du codage arithmétique par rapport au codage de Huffman.

Les méthodes utilisées pour lutter contre la sensibilité au bruit des codes arithmétiques consistent généralement à réaugmenter la redondance du train binaire, soit en introduisant un code correcteur d'erreurs, soit en insérant des marqueurs spécifiques

dans la chaîne. Dans [Elm99a], l'auteur introduit de la redondance sous forme de bits de contrôle de parité, inclus dans le processus de codage. Un intervalle de probabilité affecté à aucun symbole de la source ou bien des marqueurs insérés à des positions connues dans la séquence de symboles sont exploités pour de la détection d'erreurs dans [BCI+97,Elm99b,SCW00]. Cette capacité peut être couplée à une procédure de retransmission à la demande (ARQ) [CR00,Elm99b] ou utilisée conjointement avec un code correcteur d'erreurs [KCR98]. Le décodage séquentiel de codes arithmétiques à été étudié dans [PHS01], de façon à permettre la correction en plus de la détection d'erreurs. La complexité de cette approche a été réduite dans [DHS01] en utilisant une modulation codée en treillis combinée avec un algorithme de décodage "list Viterbi".

Dans ce chapitre, nous proposons de formaliser les dépendances et contraintes entre les variables impliquées dans le codage arithmétique par des réseaux bayésiens. Nous suivons en ce sens la méthodologie utilisée dans [GFGR01] pour les codes de Huffman. Une présentation des réseaux bayésien et des algorithmes d'estimation qui peuvent y être associés est faite en annexe B. Cette formalisation s'appuie sur des modèles d'états de la source et du codeur arithmétique. La source est modélisée par une chaîne de Markov d'ordre 1. Le codeur arithmétique s'appuie sur les probabilités conditionnelles de la source pour convertir une séquence de symboles en une séquence de bits dont la longueur dépend de la réalisation de la source. Les données reçues à la sortie du canal de transmission constituent des observations sur la séquence de bits. Retrouver la séquence de symboles qui a été transmise équivaut à déterminer la séquence d'états pris par le modèle de codeur. Comme pour tout codage à longueur variable, la segmentation de la séquence de mesures en états du modèle de codeur est aléatoire. De plus, dans le cas du codage arithmétique, certaines transitions ne produisent aucun bits et une mesure unique peut contenir une information concernant plusieurs états. Une autre difficulté provient du fait que l'arbre de codage croît exponentiellement avec le nombre de symboles codés.

Les modèles stochastiques de codeur et de décodeur posent une base qui permet de concevoir des algorithmes de décodage souple robustes aux erreurs. La complexité exponentielle induite par le modèle de dépendances sous-jacent est contrôlée par une simple technique d'élagage, qui permet de ramener cette complexité à un niveau réaliste pour une perte raisonnable de fiabilité de l'estimation. Les modèles utilisés procurent également un cadre adapté à l'introduction de *marqueurs de synchronisation*. Ces marqueurs servent de repères pour favoriser la vraisemblance des chemins correctement synchronisés dans le treillis d'estimation. Cette idée de *synchronisation souple* permet d'augmenter la capacité de resynchronisation de la chaîne pour un débit excédent contrôlé. L'algorithme d'estimation peut également être associé à un code de canal dans un schéma itératif comparable aux turbo codes en série. Après avoir validé l'algorithme d'estimation sur des sources théoriques, nous l'appliquons au codage arithmétique contextuel utilisé dans le standard JPEG2000. Nous validons ainsi l'algorithme sur des données images.

3.2 Notations

Soit $S = S_1 \dots S_K$ une séquence de symboles quantifiés prenant leurs valeurs dans un alphabet fini $\mathcal{A} = \{a_1, a_2, \dots, a_i, \dots, a_M\}$ composé de $M = 2^q$ symboles. S est codée en une séquence de bits $U = U_1 \dots U_N$ par un codeur arithmétique. k et n représentent des index temporels génériques respectivement pour *l'horloge symbole* et *l'horloge bit*. La longueur N du train binaire est une variable aléatoire fonction de S. Les longueurs K et N seront supposées connues en général. On suppose que la séquence $S = S_1 \dots S_K$ forme une chaîne de Markov d'ordre 1. Le train binaire U est transmis sur un canal sans mémoire et reçu sous forme de mesures Y. Le problème que nous considérons consiste à estimer S sachant les valeurs observées y. Nous utilisons les lettres majuscules pour désigner les variables aléatoires et les lettres minuscules pour les réalisations de ces variables. Pour désigner une suite de variables, nous utilisons la notation $X_u^v = \{X_u, X_{u+1}, \dots, X_v\}$ ou $\bar{X}_I$, avec I l'ensemble des index $\{u, u+1, \dots, v\}$.

3.3 Principe du codage arithmétique

L'idée du codage arithmétique a été tout d'abord évoquée dans le travail original de Shannon sur la théorie de l'information [Sha48]. Cette idée a été exploitée en premier par Elias et publiée dans [Abr63] au début des années 1960. Des mises en oeuvre pratiques ont été proposées par Rissanen [Ris76][Ris79] et Pasco [Pas76]. Par la suite, diverses améliorations ont été publiées. Il existe à présent de nombreuses versions d'algorithmes de codage arithmétique. Les travaux de Langdon [Lan84] et Witten [WNC87][WNC98] sont des références reconnues. Howard et Vitter ont également étudié des algorithmes à complexité réduite [HV92b][HV93]. Des algorithmes de codage arithmétique ont été adoptés dans des standards de compression. Ainsi le Q-Coder [PMLA88] est utilisé dans le standard de compression d'images binaires JBIG. Une variante de ce dernier, le MQ-Coder est utilisée dans le standard de compression d'images fixes JPEG2000 [ISO00]. Enfin, plus récemment, le CABAC (Context Adaptive Binary Arithmetic Coder) [MBHW01] a été adopté pour le standard de compression vidéo MPEG4-10/H264 [Joi02].

Nous rappelons brièvement les notions concernant le codage arithmétique qui sont utilisées par la suite en nous basant sur [Say00] et [WNC87]. Les techniques d'estimation et de synchronisation souple décrites dans ce chapitre restent valides pour tout algorithme de codage arithmétique. Nous utilisons par exemple le MQ-Coder en section 3.9.

Considérons l'exemple simple d'une source qui prend ses valeurs dans l'alphabet $\{a_1, a_2, a_3, a_4\}$ avec la distribution stationnaire $\mathbb{P}_s = [0.6\ 0.2\ 0.1\ 0.1]$. La figure 3.1 utilise cette source pour illustrer le principe du codage arithmétique. L'intervalle réel $[0, 1[$ est partitionné en quatre segments représentant les quatre symboles de l'alphabet. La taille de chaque segment est la probabilité stationnaire du symbole correspondant. La partition de l'intervalle unité, c'est à dire les bornes des segments, est donnée par les probabilités cumulées des symboles. L'intervalle correspondant au premier symbole à coder est choisi et devient l'intervalle courant. L'intervalle courant est à son tour divisé

en quatre segments, proportionnellement aux probabilités stationnaires des symboles. Si la source est modélisée par une chaîne de Markov d'ordre 1, les bornes des intervalles seront obtenues à partir de la loi de transition $\mathbb{P}(S_{l+1}|S_l)$. Dans ce cas, le codeur arithmétique s'adapte à l'entropie conditionnelle $H(S_{l+1}|S_l)$ du processus S, c'est à dire qu'il compresse l'innovation de la chaîne de Markov S. En général, le terme de *codeur arithmétique contextuel* est employé dans ce cas. Les symboles précédents utilisés pour définir la distribution du symbole à coder constituent le *contexte* de ce symbole. Dans le cas d'une chaîne de Markov d'ordre 1, le contexte du symbole S_l est le symbole S_{l-1}.

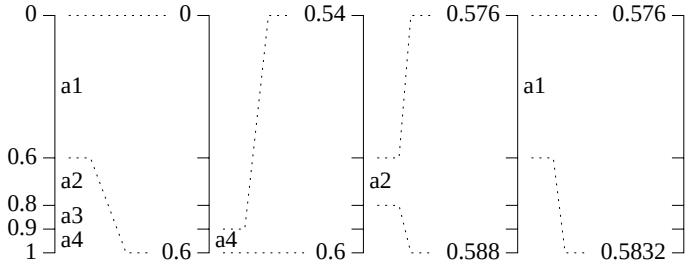


FIG. 3.1 – *Principe du codage arithmétique.*

A partir de l'exemple ci-dessus, si la séquence $a1, a4, a2, a1$ doit être codée, l'intervalle courant devient $[0.576, 0.5832[$. N'importe quel nombre appartenant à cet intervalle peut être utilisé pour identifier la séquence sans ambiguïté. Considérons par exemple 0.576. Le décodage de la séquence s'effectue en reproduisant le comportement du codeur. Tout d'abord, l'intervalle $[0, 1[$ est partitionné en fonction des probabilités cumulées de la source. Comme la valeur 0.576 appartient à l'intervalle $[0, 0.6[$, il est clair que le premier symbole codé a été $a1$. Par conséquent, le premier symbole est décodé et l'intervalle $[0, 0.6[$ est partitionné en fonction des probabilités cumulées de la source à son tour. Ce processus est répété jusqu'à ce que la séquence soit complètement décodée.

Un problème qui se pose lors de la mise en œuvre du codage arithmétique est la nécessité de disposer d'une grande précision numérique pour représenter des nombres réels très petits. Cette difficulté peut être résolue en s'appuyant sur la représentation binaire des nombres réels dans l'intervalle $[0, 1[$ [WNC87]. Tout nombre appartenant à l'intervalle $[0, 0.5[$ a son premier bit égal à 0, alors que tout nombre appartenant à l'intervalle $[0.5, 1[$ a son premier bit égal à 1. Par conséquent, pendant le processus de codage, dés que l'intervalle courant est entièrement contenu dans $[0, 0.5[$ ou $[0.5, 1[$, le bit correspondant est émis et la taille de l'intervalle est doublée. Un traitement spécifique est nécessaire pour les intervalles qui chevauchent $\frac{1}{2}$. Si l'intervalle courant chevauche $\frac{1}{2}$ et est entièrement contenu dans $[0.25, 0.75[$, il ne peut pas être identifié par un bit unique. Sa taille est tout de même doublée, sans émettre de bit, et le nombre de fois où cette opération est effectuée avant d'émettre un bit est mémorisée. Lorsqu'un intervalle pour lequel un bit $U_i = u_i$ est émis est atteint, alors ce bit est suivi par la séquence de bits $U_{i+1} = u_{i+1} \dots U_{i+n} = u_{i+1}$, avec n le nombre d'opération de remise

à l'échelle effectuées avant l'émission de U_i et $u_{i+1} = u_i + 1 \bmod 2$. L'utilisation de cette technique garantit que l'intervalle courant satisfait en permanence $low < 0.25 < 0.5 \leq high$ ou $low < 0.5 < 0.75 \leq high$, avec low et $high$ les bornes inférieures et supérieures de l'intervalle courant respectivement. Ainsi on évite le problème de précision pouvant survenir lorsque la séquence de symboles codée maintient l'intervalle courant autour de $\frac{1}{2}$. Un traitement plus détaillé ainsi que des techniques permettant de réduire la complexité du codage arithmétique peuvent être trouvé dans [Lan84], [HV92a], [HV92b], [HV94], [WNC98] et [Say00].

Un dernier point important concerne la terminaison du processus de codage arithmétique. Il faut en effet s'assurer au moment du codage que le décodeur pourra identifier la séquence sans ambiguïté jusqu'au dernier symbole S_K. Cette question est mise en avant dans [WNC87], où il est dit qu'il est nécessaire de terminer le train binaire par un ensemble de bits adéquat qui assure que ce train binaire identifie uniquement l'intervalle correspondant au dernier symbole S_K. L'utilisation de la stratégie de remise à l'échelle décrite plus haut impose que l'intervalle courant chevauche soit l'intervalle $]0.25, 0.5]$, soit l'intervalle $]0.5, 0.75]$. Le train binaire se termine alors respectivement soit par un 0 suivi d'une séquence de 1, soit par un 1 suivi d'une séquence de 0. Le nombre de bits nécessaires pour compléter le train binaire est donné par $\lceil -log_2(high - low) + n_{scl} \rceil$, avec $high$ et low respectivement les bornes supérieures et inférieures de l'intervalle courant et n_{scl} le nombre de remises à l'echelle qui ont été effectuées depuis la dernière émission de bit.

La question de la terminaison du processus de codage permet de remarquer que le codeur et le décodeur ne sont pas synchronisés. Ils effectuent deux conversions différentes entre horloge bit et horloge symbole. On peut reformuler cette remarque de la manière suivante : si après avoir codé les k premiers symboles d'une séquence S le codeur à généré n bits, et si après avoir décodés ces n premier bits le décodeur à généré k' symboles, alors on ne peut pas affirmer que $k = k'$.

3.4 Modélisation

Dans cette section, nous nous efforçons d'analyser les dépendances entre les variables impliquées dans les processus de codage et de décodage arithmétique. Des modèles de dépendances peuvent être obtenus en considérant soit l'arbre m-aire de codage, soit l'arbre binaire de décodage. Nous considérons que la source est stationnaire et constitue une chaîne de Markov d'ordre 1.

3.4.1 Modèle à horloge symbole du codeur

Etant donné l'alphabet $\mathcal{A}$ de taille M, la séquence de symboles S_1^K est convertie en une séquence de bits par un arbre de décision M-aire. Cet arbre de codage peut être vu comme un automate qui modélise la distribution du train binaire. Le codage d'un symbole détermine le choix d'une branche dans l'arbre. Chaque nœud de l'arbre correspond à un état X du codeur arithmétique, auquel peut être associée l'émission d'une séquence de bits de longueur variable. Les transitions successives dans l'arbre

suivent la distribution de la source ($\mathbb{P}(S_k|S_{k-1})$ pour une chaîne de Markov d'ordre 1). Le codeur arithmétique effectue une conversion d'horloge symbole vers horloge bit qui dépend de la réalisation de la source. Le nombre de bits produit par chaque transition du modèle est aléatoire. Par conséquent, la structure de dépendances entre la séquence de mesures sur les bits et les états de l'arbre de codage est aléatoire. Afin de "capturer" cette structure de dépendances, nous considérons le processus de Markov augmenté $(X,N) = (X_1,N_1)\dots(X_k,N_k)\dots(X_K,N_K)$, où $k = 1\dots K$ représente l'horloge symbole. La quantité N_k est le nombre de bits émis quand k symboles ont été codés et X_k est l'état interne du codeur arithmétique. Ainsi les transitions successives dans l'arbre de codage correspondent à des transitions entre (X_{k-1},N_{k-1}) et (X_k,N_k). La taille de l'arbre croît exponentiellement avec le nombre de symboles codés. Les feuilles de l'arbre correspondent à toutes les combinaisons de symboles qui peuvent être codées. La valeur exacte de X_k dépend de la mise en oeuvre du codeur arithmétique. Elle doit au moins contenir de quoi spécifier un segment de l'intervalle $[0,1[$.

Le processus de codage arithmétique démarre à l'état initial (X_0,N_0), avec X_0 l'intervalle unité et $N_0 = 0$. Dans l'exemple simple de la section 3.3, si le premier symbole codé est a_1, alors la branche qui conduit à l'état (X_1,N_1) est choisie, avec $X_1 = [0,0.6[$ et $N_1 = 0$. Si le second symbole codé est a_4, alors l'état suivant est (X_2,N_2) avec $X_2 = [0.54,0.6[$. Ce nouvel intervalle permet de déterminer que les trois premiers bits sont 100, par conséquent $N_2 = 3$.

Pour éviter les problèmes de précision numérique lors de la mise en oeuvre, nous utiliserons à présent la technique de remise à l'échelle décrite en section 3.3 [WNC87]. La variable d'état X_k est alors composée des trois variables l_k, h_k et $nscl_k$, avec $[l_k,h_k[$ l'intervalle courant et $nscl_k$ le nombre de remises à l'échelle qui ont été effectuées depuis la dernière émission de bit. Ainsi l'état initial est défini par $l_0 = 0$, $h_0 = 1$ et $nscl_k = 0$. Les actions qui sont effectuées à chaque transition de (X_k,N_k) vers (X_{k+1},N_{k+1}), déclenchées par le symbole $S_{k+1} = a_i$, sont décrites dans le tableau 3.1. Les quantités $P_{cum}(a_i) = \sum_{t=1}^{i} P(a_t)$ et $P_{cum}(a_i|a_j) = \sum_{t=1}^{i} P(a_t|a_j)$ représentent respectivement les probabilités cumulées stationnaires et conditionnelles des symboles de l'alphabet $\mathcal{A}$. Il est important de rappeler que chaque état X_k dépend de la totalité de la séquence S_1^k, d'où la structure d'arbre. A chaque état (X_k,N_k) est associé l'émission d'une séquence de bits $\bar{U}_k = U_{N_{k-1}+1}^{N_k}$ de longueur $N_k - N_{k-1}$, avec $N_k \geq N_{k-1}$. Les mesures $\bar{Y}_k$ des bits $\bar{U}_k$ sont obtenues à la sortie du canal de transmission. Nous aboutissons ainsi à la structure de dépendances représentées graphiquement par la figure 3.2. Nous avons donc un modèle de Markov caché à horloge symbole du codeur arithmétique. On remarque que l'ensemble des valeurs possibles des états (X_k,N_k) varie en fonction de k.

3.4.2 Modèle à horloge bit du décodeur

De la même manière que pour le codeur, on peut construire un modèle à horloge bit du décodeur. Une séquence de bits U_1^n codés arithmétiquement peut être convertie en une séquence de symboles $S_1^{K_n}$ par un arbre de décision binaire. Chaque bit détermine

1. Initialiser (X_{k+1}, N_{k+1}) : soit $S_k = a_j$ et $S_{k+1} = a_i$.
   - $\boldsymbol{l_{k+1}} = \boldsymbol{l_k} + (\boldsymbol{h_k} - \boldsymbol{l_k})\mathbb{P}_{cum}(a_{i-1}|S_k = a_j)$ (si $k = 0$, $\mathbb{P}_{cum}(a_{i-1})$ est utilisé).
   - $\boldsymbol{h_{k+1}} = \boldsymbol{l_k} + (\boldsymbol{h_k} - \boldsymbol{l_k})\mathbb{P}_{cum}(a_i|S_k = a_j)$ (si $k = 0$, $\mathbb{P}_{cum}(a_i)$ est utilisé).
   - $\boldsymbol{nscl_{k+1}} = \boldsymbol{nscl_k}$.
   - $N_{k+1} = N_k$.
2. Tant qu'une des trois mises à l'échelle est possible, mettre à jour (X_{k+1}, N_{k+1}) :
   - Si $\boldsymbol{h_{k+1}} < 0.5$:
     - $\boldsymbol{l_{k+1}} = 2\boldsymbol{l_{k+1}}$.
     - $\boldsymbol{h_{k+1}} = 2\boldsymbol{h_{k+1}}$.
     - Emettre un bit 0.
     - Emettre $\boldsymbol{nscl_{k+1}}$ bits 1.
     - $N_{k+1} = N_{k+1} + 1 + \boldsymbol{nscl_{k+1}}$.
     - $\boldsymbol{nscl_{k+1}} = 0$.
   - Si $0.5 \leq \boldsymbol{l_{k+1}}$:
     - $\boldsymbol{l_{k+1}} = 2(\boldsymbol{l_{k+1}} - 0.5)$.
     - $\boldsymbol{h_{k+1}} = 2(\boldsymbol{h_{k+1}} - 0.5)$.
     - Emettre un bit 1.
     - Emettre $\boldsymbol{nscl_{k+1}}$ bits 0.
     - $N_{k+1} = N_{k+1} + 1 + \boldsymbol{nscl_{k+1}}$.
     - $\boldsymbol{nscl_{k+1}} = 0$.
   - Si $0.25 \leq \boldsymbol{l_{k+1}} < 0.5 \leq \boldsymbol{h_{k+1}} < 0.75)$:
     - $\boldsymbol{l_{k+1}} = 2(\boldsymbol{l_{k+1}} - 0.25)$.
     - $\boldsymbol{h_{k+1}} = 2(\boldsymbol{h_{k+1}} - 0.25)$.
     - $\boldsymbol{nscl_{k+1}} = \boldsymbol{nscl_{k+1}} + 1$.
3. Fin : (X_{k+1}, N_{k+1}) est à jour, $N_{k+1} - N_k$ bits ont été émis.

TAB. 3.1 – *Actions déclenchées par les transitions du codeur arithmétique.*

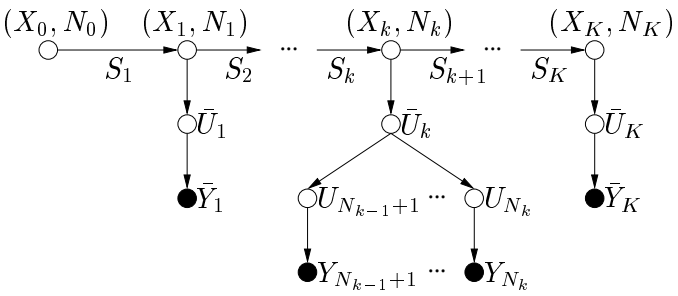


FIG. 3.2 – *Modèle à horloge symbole du codeur arithmétique ($N_k \geq N_{k-1}$). Les points blancs et noirs représentent respectivement les variables cachées et observées.*

le choix d'une branche de l'arbre. Chaque nœud de l'arbre corrspond à une séquence de bits U_1^{n-1} à partir de laquelle deux transitions $U_n = 0$ ou $U_n = 1$ sont possibles. A chaque nœud de l'arbre est associé un état X_n du décodeur arithmétique . A chacun de ces états peut être associé un nombre variable de symboles décodés. Ce nombre de symboles décodés dépend de la réalisation de la séquence de bit et est, par conséquent, aléatoire. Nous capturons cette structure de dépendances par le processus de Markov Augmenté (X_n, K_n), avec X_n l'état interne du décodeur arithmétique, K_n le nombre maximum de symboles qui peuvent être décodés à cet instant et $n = 1 \dots N$ les index d'horloge bit. Ainsi, contrairement au codeur, le décodeur arithmétique est synchronisé sur l'horloge bit. Les transitions entre (X_{n-1}, K_{n-1}) et (X_n, K_n) suivent la distribution des symboles décodés correspondant : $\mathbb{P}(S_{K_{n-1}+1} \dots S_{K_n} | S_1^{K_{n-1}})$. Dans le cas d'une chaîne de Markov d'ordre 1, on a $\mathbb{P}(S_{K_{n-1}+1}^{K_n} | S_1^{K_{n-1}}) = \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k | S_{k-1})$. Nous aboutissons ainsi à la structure de dépendances décrite graphiquement par la figure 3.3. A chaque état (X_n, K_n) du décodeur est associé une séquence de symboles décodés $S_{K_{n-1}+1}^{K_n}$. La transition entre deux états (X_{n-1}, K_{n-1}) et (X_n, K_n) est déclenchée par le bit U_n, dont l'observation Y_n est disponible.

Les actions effectuées à chaque transition sont résumées dans le tableau 3.2. L'état X_n du décodeur arithmétique contient deux intervalles : $[\boldsymbol{l_n}, \boldsymbol{h_n}[$ l'intervalle défini par les n bits reçus et $[\boldsymbol{l_{K_n}}, \boldsymbol{h_{K_n}}[$ l'intervalle correspondant aux K_n symboles pouvant être décodés sachant $[\boldsymbol{l_n}, \boldsymbol{h_n}[$. A chaque symbole décodé, l'intervalle $[\boldsymbol{l_{K_n}}, \boldsymbol{h_{K_n}}[$ est mis à jour de la même manière que lors du codage. Lorsqu'une remise à l'échelle est possible, elle est appliquée sur les deux intervalles. Il est inutile de conserver le nombre de remises à l'échelle effectuées quand $[\boldsymbol{l_{K_n}}, \boldsymbol{h_{K_n}}[$ chevauche $\frac{1}{2}$. Les deux intervalles sont initialisés à $[0, 1[$.

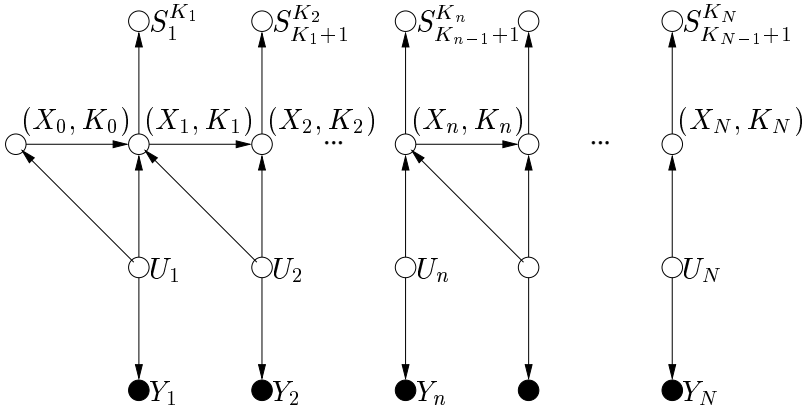


FIG. 3.3 – *Modèle à horloge bit du décodeur arithmétique. Les points blancs et noirs représentent respectivement les variables cachées et observées.*

---

1. Initialiser (X_{n+1}, K_{n+1}) : soit $S_{K_n} = a_j$.
   - $K_{n+1} = K_n$.
   - $\boldsymbol{l_{K_{n+1}}} = \boldsymbol{l_{K_n}}$ et $\boldsymbol{h_{K_{n+1}}} = \boldsymbol{h_{K_n}}$.
   - Si $u_{n+1} = 0$
     $\boldsymbol{l_{n+1}} = \boldsymbol{l_n}$ et $\boldsymbol{h_{n+1}} = \frac{1}{2}(\boldsymbol{l_n} + \boldsymbol{h_n})$.
     Sinon
     $\boldsymbol{l_{n+1}} = \frac{1}{2}(\boldsymbol{l_n} + \boldsymbol{h_n})$ et $\boldsymbol{h_{n+1}} = \boldsymbol{h_n}$.
2. Chercher $a_i \in \mathcal{A}$ tel que
   - $L = \boldsymbol{l_{K_{n+1}}} + (\boldsymbol{h_{K_{n+1}}} - \boldsymbol{l_{K_{n+1}}}) P_{cum}(a_{i-1}|S_{K_{n+1}})$ (si $K_{n+1} = 0$, $P_{cum}(a_{i-1})$ est utilisé),
   - $H = \boldsymbol{l_{K_{n+1}}} + (\boldsymbol{h_{K_{n+1}}} - \boldsymbol{l_{K_{n+1}}}) P_{cum}(a_i|S_{K_{n+1}})$ (si $K_{n+1} = 0$, $P_{cum}(a_i)$ est utilisé),
   - $L \leq l_{n+1} < h_{n+1} < H$ (condition pour que le symbole a_i puisse être décodé),
3. Si a_i existe, aller en 4, sinon aller en 5.
4. mettre à jour (X_{n+1}, K_{n+1}) :
   - $K_{n+1} = K_{n+1} + 1$,
   - $S_{K_{n+1}} = a_i$,
   - $\boldsymbol{l_{K_{n+1}}} = L$,
   - $\boldsymbol{h_{K_{n+1}}} = H$,
   - Remettre à l'échelle $[\boldsymbol{l_{n+1}}, \boldsymbol{h_{n+1}}]$ et $[\boldsymbol{l_{K_{n+1}}}, \boldsymbol{h_{K_{n+1}}}]$ si nécessaire,
   - Aller en 2.
5. Fin : (X_{n+1}, K_{n+1}) est à jour, $K_{n+1} - K_n$ symboles ont été décodés.

TAB. 3.2 – *Actions déclenchées par chaque transition du décodeur arithmétique.*

3.5 Estimation

3.5.1 Estimation à horloge symbole

L'estimation des états cachés $(X,N) = (X_1,N_1)\dots(X_K,N_K)$ à horloge symbole (figure 3.2) est équivalente à l'estimation de la séquence de transitions entre ces états, c'est à dire à l'estimation de la séquence $S = S_1 \dots S_k \dots S_K$, sachant les observations Y_1^N à la sortie du canal. Cela peut s'exprimer sous la forme

$$\begin{aligned} \mathbb{P}(S_1^K|Y_1^N) &= \mathbb{P}(S_1^K)\cdot\mathbb{P}(U_1^N|Y_1^N) \\ &\propto \mathbb{P}(S_1^K)\cdot\mathbb{P}(Y_1^N|U_1^N), \end{aligned} \tag{3.1}$$

ou $\propto$ désigne un facteur de normalisation. Comme les bits émis par un codeur arithmétique sont distribués uniformément et indépendamment, ce facteur de normalisation est $\frac{\mathbb{P}(U_1^N)}{\mathbb{P}(Y_1^N)} = 1$. La quantité $\mathbb{P}(S_1^K|Y_1^N)$ peut être calculée en une passe avant unique grâce à la décomposition

$$\begin{aligned} \mathbb{P}(S_1^k|Y_1^{N_k}) &\propto \mathbb{P}(S_1^{k-1}|Y_1^{N_{k-1}})\cdot\mathbb{P}(S_k|S_{k-1})\cdot \\ &\quad \mathbb{P}(Y_{N_{k-1}+1}^{N_k}|Y_1^{N_{k-1}},U_1^{N_k}), \end{aligned} \tag{3.2}$$

ou N_k est le nombre total de bits émis lorsqu'on est dans l'état X_k. En considérant une source de Markov d'ordre 1 et un canal sans mémoire, cette probabilité *a posteriori* peut être réécrite

$$\begin{aligned} \mathbb{P}(S_1^k|Y_1^{N_k}) &\propto \mathbb{P}(S_1^{k-1}|Y_1^{N_{k-1}})\cdot\mathbb{P}(S_k|S_{k-1})\cdot \\ &\quad \mathbb{P}(Y_{N_{k-1}+1}^{N_k}|U_{N_{k-1}+1}^{N_k}) \end{aligned} \tag{3.3}$$

Le calcul est initialisé avec $\mathbb{P}(S_1 = s_1|Y_1^{N_1=n_1} = y_1^{N_1=n_1})$, pour les M valeurs possibles de s_1. Ces probabilités sont données par

$$\begin{aligned} \mathbb{P}(S_1 = s_1|Y_1^{N_1=n_1}) &\propto \mathbb{P}(S_1 = s_1)\cdot \\ &\quad \mathbb{P}(Y_1^{N_1} = y_1^{n_1}|U_1^{N_1} = u_1^{n_1}), \end{aligned} \tag{3.4}$$

ou le premier terme est donné par la distribution stationnaire de la source et le second terme modélise le bruit introduit par le canal sans mémoire. Ce second terme est donné par

$$\begin{aligned} &\mathbb{P}(Y_{N_{k-1}+1}^{N_k} = y_{n_{k-1}+1}^{n_k}|U_{N_{k-1}+1}^{N_k} = u_{n_{k-1}+1}^{n_k}) = \\ &\begin{cases} \displaystyle\prod_{n=N_{k-1}+1}^{N_k} \mathbb{P}(Y_n = y_n|U_n = u_n) \textbf{ si } N_k > N_{k-1} \\ 1 \textbf{ sinon.} \end{cases} \end{aligned} \tag{3.5}$$

Partant de cette initialisation, l'équation 3.3 est utilisée pour calculer $\mathbb{P}(S_1^k|Y_1^{N_k})$ pour chaque instant symbole consécutivement. A la fin de l'estimation, les marginales a posteriori $\mathbb{P}(S_1^K = s_1^K|Y_1^{N_K})$ sont connues pour toutes les séquences s_1^K possibles.

Comme nous considérons toutes les séquences de symboles possibles, la taille du treillis d'estimation croît exponentiellement avec le nombre de symboles K (complexité en $O(M^K)$). La complexité de l'estimation est maintenue à un niveau réaliste grâce à la méthode d'élagage décrite en section 3.6. Finalement, la structure de l'algorithme incluant l'élagage est résumée sur la figure 3.4. L'arbre de codage est construit niveau par niveau, ce qui correspond à une stratégie dite "en largeur" (ou "breadth first" dans [PHS01]). D'autres stratégies sont possibles, mais nous ne les considérons pas ici.

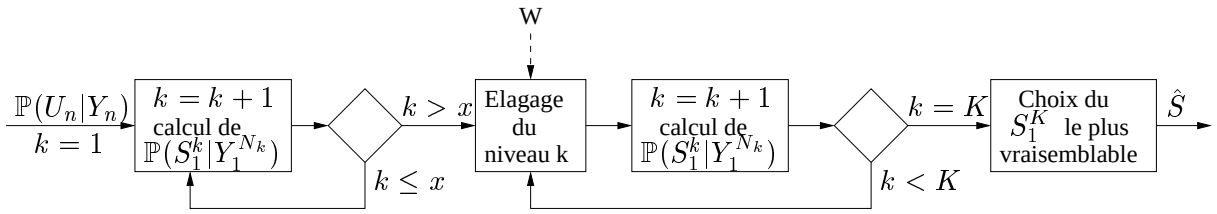


FIG. 3.4 – *Procédé d'estimation séquentielle à horloge symbole.*

3.5.2 Estimation à horloge bit

L'estimation peut également se baser sur le modèle à horloge bit décrit par la figure 3.3. La passe avant unique permet de calculer $\mathbb{P}(S_1^{K_n}|Y_1^n)$, où K_n est une variable aléatoire désignant le nombre total de symboles décodés à la réception du bit n. Le terme $\mathbb{P}(S_1^{K_n}|Y_1^n)$ peut s'exprimer

$$\mathbb{P}(S_1^{K_n}|Y_1^n) \quad \propto \quad \mathbb{P}(S_1^{K_n}) \cdot \mathbb{P}(Y_1^n|U_1^n) \tag{3.6}$$

Là encore, en considérant une source de Markov d'ordre 1, on a

$$\mathbb{P}(S_1^{K_n}) = \mathbb{P}(S_1^{K_{n-1}}) \cdot \mathbb{P}(S_{K_{n-1}+1}^{K_n}|S_1^{K_{n-1}}), \tag{3.7}$$

avec

$$\mathbb{P}(S_{K_{n-1}+1}^{K_n}|S_1^{K_{n-1}}) = \begin{cases} \displaystyle\prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k|S_{k-1}) \textbf{ si } K_n > K_{n-1} \\ 1 \textbf{ sinon.} \end{cases} \tag{3.8}$$

Pour un canal sans mémoire, l'équation 3.6 peut se réécrire

$$\begin{aligned} \mathbb{P}(S_1^{K_n}|Y_1^n) \quad &\propto \quad \mathbb{P}(S_1^{K_{n-1}}) \cdot \mathbb{P}(Y_1^{n-1}|U_1^{n-1}) \cdot \\ &\quad\ \mathbb{P}(Y_n|U_n) \cdot \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k|S_{k-1}) \\ &\propto \quad \mathbb{P}(S_1^{K_{n-1}}|Y_1^{n-1}) \cdot \mathbb{P}(Y_n|U_n) \cdot \\ &\quad\ \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k|S_{k-1}) \end{aligned} \tag{3.9}$$

On remarque que lorsque l'estimation est basée sur le modèle à horloge symbole, le nombre de symboles K doit être connu, mais pas le nombre de bits N. Dans le cas du modèle à horloge bit, au contraire, N doit être connu et K peut être estimé. La structure de l'algorithme d'estimation à horloge bit est résumée sur la figure 3.5.

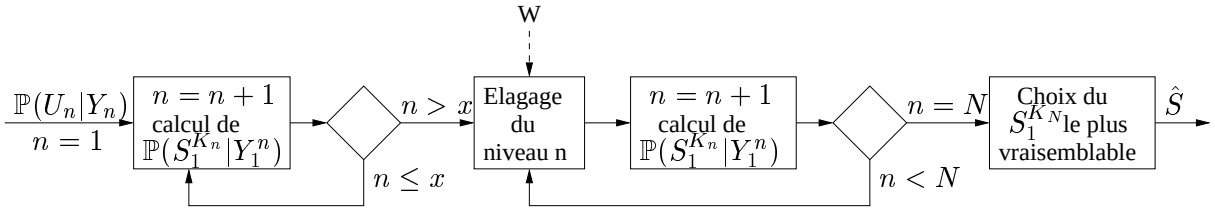


FIG. 3.5 – *Procédé d'estimation séquentielle à horloge bit.*

3.6 Elagage

Comme le nombre d'états du treillis d'estimation croît exponentiellement avec le nombre de symboles codés, nous considérons des techniques d'élagage afin de contrôler la complexité de l'estimation. Tout d'abord, le fait de connaître à la fois le nombre de bits N et le nombre de symboles K constitue une contrainte sur la terminaison de l'estimation. Cela permet d'éliminer les chemins qui ne respectent pas cette contrainte. Cette réduction de la complexité reste cependant insuffisante.

Le critère d'élagage principal que nous introduisons est basé sur un seuil appliqué sur la vraisemblance des chemins. La vraisemblance d'un chemin est donnée par les équations 3.1 (horloge symbole) ou 3.6 (horloge bit). Dans ces deux équations, les termes $\mathbb{P}(Y|U)$ sont liés au modèle de canal. $\mathbb{P}(Y_n = y_n|U_n = u_n)$ représente la probabilité qu'un bit y_n soit observé sachant qu'un bit u_n a été transmis, ce qui équivaut à la probabilité que le bit observé soit erroné. Soit p la probabilité d'erreur binaire moyenne que l'on accepte de tolérer sur le canal durant l'estimation. A chaque instant bit n (N_k à horloge symbole), on supprime tous les chemins du treillis d'estimation pour lesquels $\mathbb{P}(Y_1^n|U_1^n) < (1-p)^n$. Cela revient à ne pas considérer les chemins pour lesquels la probabilité d'erreur binaire moyenne à la sortie du canal est supérieure à p.

Enfin, pour borner la complexité de l'estimation, nous ajoutons une contrainte fixe sur le nombre de nœuds du treillis. A chaque étape du processus d'estimation, seuls les W nœuds les plus vraisemblables sont conservés. Le choix de W dépend des ressources disponibles. Afin de garantir une terminaison correcte de l'estimation, nous raffinons cette contrainte en conservant les w nœuds les plus vraisemblables pour chaque valeur possible de N_k (horloge symbole) ou K_n (horloge bit). Dans ce cas, on a, à horloge symbole, $W = (N_{kmax} - N_{kmin} + 1)w$, ou N_{kmax} et N_{kmin} dépendent respectivement de la probabilité de la séquence la plus et la moins vraisemblable. De même, à horloge bit, on a $W = (K_{nmax} - K_{nmin} + 1)w$. Afin d'assurer la significativité des mesures de vraisemblance, nous n'appliquons les technique d'élagage qu'à partir d'un niveau $n > x$ ($k > x$ à horloge symbole).

Du fait de l'utilisation de l'élagage, l'algorithme d'estimation proposé s'apparente au *décodage séquentiel*. La différence entre notre algorithme d'estimation et le décodage séquentiel classique réside principalement dans la modélisation. Celle-ci nous permet de manipuler le codeur arithmétique et de contrôler l'élagage, mais aussi d'exploiter un ajout de redondance supplémentaire, comme cela est expliqué ci-dessous en section 3.8. L'idée d'appliquer le décodage séquentiel au décodage robuste de codes arithmétiques a été proposée seulement récemment dans [PHS01]. Pour un traitement complet du décodage séquentiel, se reporter à [VK79].

3.7 Codage arithmétique adaptatif

Jusqu'à présent, nous avons considéré que la distribution de la source était stationnaire. Un des avantages du codage arithmétique par rapport aux codes de Huffman est de pouvoir adapter dynamiquement le modèle de source au fur et à mesure que les symboles sont lus et codés. Là où un codeur de Huffman nécessite de reconstruire l'arbre de codage chaque fois que les statistiques de la source sont mises à jour, un codeur arithmétique s'adapte naturellement lors de l'étape de subdivision de l'intervalle courant. On parle alors de codage arithmétique adaptatif. Les avantages sont évidemment de pouvoir s'adapter à des sources non stationnaires, mais aussi de ne pas avoir besoin de modèle de source connu a priori ou calculé puis transmis avec les données. Les inconvénients sont l'inefficacité du codeur à l'initialisation et une complexité accrue. Cette dernière provient de l'étape de mise à jour du modèle de source qui peut intervenir à chaque symbole codé. La perte en compression lors de l'initialisation du processus de codage est compensée par le fait de ne pas avoir à transmettre de modèle statistique avec les données compressées.

Nous ne développons pas ici le fonctionnement du codage arithmétique adaptatif. Il est expliqué en détail dans [WNC87] et [WNC98]. Le Q-Coder [PMLA88] et le CABAC [MBHW01][Joi02] sont des exemples de codeurs adaptatifs. Quelle que soit la technique utilisée, le codage arithmétique adaptatif requiert l'utilisation d'une structure de données pour stocker les statistiques de la source et les mettre à jour dynamiquement. Ces statistiques dépendent de la réalisation de la source, donc du chemin suivi dans l'arbre de codage ou de décodage. Par conséquent, pour pouvoir appliquer l'algorithme d'estimation décrit précédemment (section 3.5), il suffit d'inclure la structure de données contenant les statistiques dans l'état X_k ou X_n respectivement du codeur ou du décodeur arithmétique.

Considérons par exemple le cas de l'estimation à horloge symbole. L'état X_k du codeur arithmétique est défini par les trois variables l_k, h_k et $nscl_k$. supposons que la source est M-aire et est distribuée indépendamment. Un tableau d'entier F de taille M permet de compter le nombre d'occurences des symboles. A un instant k, le tableau est dans l'état F_k. L'état X_k du codeur arithmétique adaptatif est maintenant défini par les quatres variables l_k, h_k, $nscl_k$ et F_k. A chaque symbole codé, F_k est mis à jour. L'algorithme d'estimation à horloge symbole vu en section 3.5.1 s'applique sans difficulté. Le tableau F peut être initialisé soit avec un modèle a priori de la source,

soit avec un modèle uniforme si on ne dispose strictement d'aucune information sur la source. Généralement, dans ce genre de méthode, le nombre d'occurences des symboles qui est mémorisé dans le tableau F est remis à l'échelle périodiquement, de façon à réduire l'influence du passé et ainsi à favoriser l'adaptation aux non-stationnarités de la source.

Considérons à présent que la source est une chaîne de Markov d'ordre 1. Dans ce cas, les occurences des symboles doivent être comptées conditionnellement au symbole précédent. La taille du tableau F devient alors M^2. Le maintien de cette structure de données additionnelle par état X du treillis d'estimation implique un coût supplémentaire en complexité et en consommation mémoire. D'une manière générale, le codage arithmétique adaptatif est plus coûteux que le codage arithmétique simple. Les méthodes intégrées dans certains standards de compression, telles que le Q-coder ou le CABAC, sont conçues de manière à limiter ce coût au maximum. Dans le cas du CABAC, par exemple, une modélisation contextuelle de la source où le nombre de contextes est volontairement limité à 126 est adoptée. Ainsi, un bon compromis entre la complexité et la fiabilité du modèle de source est obtenu. On peut envisager d'appliquer l'algorithme d'estimation proposé dans un tel contexte. On s'attend cependant à une moins grande robustesse que pour des sources stationnaires dont le modèle est connu a priori. En effet, si le modèle de source fait partie lui aussi des variables qui sont estimées, on peut craindre dans certains cas une dérive du modèle estimé par rapport au modèle réel. La qualité du modèle de source utilisé pour initialiser le codage adaptatif peut alors avoir un impact important sur la fiabilité de l'estimation.

3.8 Amélioration des performances par ajout de redondance

La robustesse apportée par les techniques de décodage souple est significative, mais insuffisante pour des taux d'erreur importants. Il est donc nécessaire d'envisager d'autres moyens de protection des données pour augmenter cette robustesse. Cela implique un ajout de redondance dans les données compressées. Cet ajout de redondance peut se faire indépendamment du décodage souple. on peut par exemple utiliser des codes correcteurs d'erreur. Il nous paraît plus intéressant de concevoir des schémas de codage où la redondance ajoutée peut être exploitée par l'algorithme de décodage souple, ceci dans le but d'atteindre de meilleurs compromis compression/robustesse.

3.8.1 Marqueurs de synchronisation

La contrainte de terminaison mentionnée en section 3.6 peut être vue comme un moyen de forcer la synchronisation à la fin de la séquence. En effet, soit elle impose au décodeur de produire le bon nombre de symbole K après avoir décodé les N bits (estimation à horloge bit), soit elle assure que les K symboles estimés produisent N bits (estimation à horloge symbole). Cette contrainte procure une observation parfaitement fiable sur le dernier état du modèle. Elle assure la synchronisation à la fin de la

séquence, mais pas au milieu de cette séquence. Il est possible d'introduire une information complémentaire spécifique pour favoriser la synchronisation sur l'ensemble de la séquence. Pour cela, nous proposons l'introduction de *marqueurs de synchronisation* à des positions connues a priori.

Ces marqueurs de synchronisation peuvent prendre la forme soit de symboles spécifiques, dans l'esprit des techniques décrites dans [BCI+97], [DHS01], [PHS01], [Elm99a] et [SCW00], soit de séquences de bits particulières, ou "marqueurs binaires", insérés respectivement dans le train de symboles et dans le train binaire à des *positions connues à horloge symbole*. Les marqueurs insérés dans la séquence de symboles sont codés arithmétiquement. La fréquence d'insertion et la longueur des marqueurs binaires sont réglés arbitrairement, suivant le niveau de redondance désiré. Il est important de distinguer les marqueurs de synchronisation des marqueurs utilisés habituellement dans les standards de compression. Ces derniers sont des mots longs qui identifient le début et la fin des segments de bits d'information. Contrairement à ceux-ci, les marqueurs de synchronisation ne constituent pas des observations parfaitement fiables. Par conséquent ils ne constituent pas des "points de synchronisation parfaite" comme la contrainte de terminaison. La probabilité de reconnaitre les marqueurs de synchronisation est inférieure à 1, d'où le nom de "marqueurs de synchronisation souple". Comme ils sont placés à des positions connues à horloge symbole, leur position dans le train binaire est aléatoire.

Les modèles et algorithmes présentés précédemment doivent tenir compte de cette information supplémentaire. L'insertion d'un symbole particulier à une position connue dans la séquence de symboles correspond à l'addition d'une section dans l'arbre M-aire avec des transitions déterministes. La présence de cette information a priori peut être exploitée par l'estimation soit à horloge symbole, soit à horloge bit. Dans le cas de l'estimation à horloge symbole, l'information apportée par la section supplémentaire de l'arbre M-aire est directement prise en compte dans l'équation 3.3, par l'intermédiaire du terme $\mathbb{P}(S_k|S_{k-1})$. De même, à horloge bit, cette information est exploitée grâce à l'équation 3.7. Si ces marqueurs sont insérés tous les f symboles, le surcoût de codage induit est donné par $\frac{-log_2(\mathbb{P}(marqueur))}{f}$ bits par symbole [DHS01]. Lorsqu'un marqueur est inséré entre deux symboles, les probabilités conditionnelles de la chaîne de Markov d'ordre 1 sont modifiées. En particulier, le symbole qui suit le marqueur perd la mémoire du symbole précédent. Comme la position du marqueur est connue a priori, ce problème peut être évité par un traitement particulier appliqué autour de chaque marqueur.

Considérons à présent l'insertion de séquences de bits spécifiques dans le train binaire. Supposons qu'un ensemble de l bits doit être ajouté au train binaire à l'instant k à horloge symbole. L'insertion de ces l bits revient à terminer la séquence de bits émise lors du codage du symbole S_k par un suffixe connu a priori: $\bar{U}_{K_n} \Rightarrow \bar{U}_{K_n} B_1 \dots B_l$. Dans le cas du modèle de codeur, les transitions de l'arbre M-aire ne sont pas modifiées. En revanche, le suffixe est ajouté aux bits émis par les états correspondant à l'instant k. Cette situation est illustrée par la figure 3.8.1. Dans le cas du modèle de décodeur, l'arbre binaire de décodage est étendu de façon à ce que tout nœud pour lequel $K_n = k$ soit suivi par le suffixe attendu. Dans cette partie supplémentaire de l'arbre binaire, les transitions sont déterministes. Les symboles pour S_k sont considérés comme décodés quand la fin du suffixe est atteinte. On remarque que l'insertion de ces marqueurs

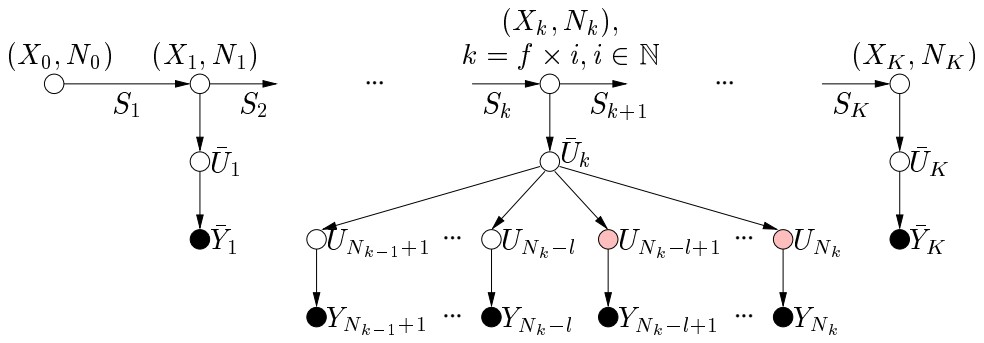


FIG. 3.6 – *Modèle du codeur arithmétique avec des marqueurs de synchronisation binaires de longueur l ajoutés tous les f symboles. Les points gris sont les bits de synchronisation. Les points blancs et noirs sont respectivement les variables cachées et observées.*

n'affecte pas la mémoire de la chaîne de Markov d'ordre 1. Les marqueurs binaires sont exploités lors de l'estimation respectivement dans l'équation 3.5 à horloge symbole et l'équation 3.9 à horloge bit, par l'intermédiaire du terme $P(Y_n = y_n|U_n = u_n)$. Dans la section 3.3 nous avons signalé que le codeur et le décodeur arithmétique sont désynchronisés. Par conséquent, si l'insertion des marqueurs est effectuée par le codeur, suivant le modèle à horloge symbole, l'estimation devra être effectuée à horloge symbole également. Pour pouvoir effectuer l'estimation à horloge bit, il est nécessaire d'insérer les marqueurs suivant ce même modèle de décodeur à horloge bit. Lors de l'estimation, on a peu de chance d'observer les marqueurs binaires attendus sur les chemins désynchronisés, ce qui a pour effet de diminuer la vraisemblance de ces chemins. Les chemins désynchronisés ont par conséquent plus de chance d'être élagués. Ainsi, les marqueurs de synchronisation permettent non seulement de favoriser les chemins correctement synchronisés, mais en plus de limiter la complexité de l'algorithme. Le surcoût induit par l'addition de marqueurs binaires de longueur l tous les f symboles est donné par $\frac{l}{f}$ bits par symboles.

Un dernier point à considérer concerne le choix des marqueurs. L'approche la plus efficace consiste à choisir des marqueurs qui dépendent de la réalisation de la source. Cette observation a déja été faite dans [Elm99b], dans le contexte de la détection d'erreur. Nous utilisons la terminaison du codage arithmétique pour définir les marqueurs binaires. Si à l'instant k à horloge symbole un marqueur doit être ajouté, nous utilisons les bits qui auraient été générés par l'étape de terminaison du code arithmétique si le symbole S_k avait été le dernier symbole de la séquence (section 3.3). Toujours dans le cas des marqueurs binaires, nous disposons de deux paramètres pour régler la redondance : la longueur des marqueurs et leur fréquence d'apparition. Dans la mesure où ces marqueurs permettent également de limiter la complexité de l'algorithme en favorisant l'élagage, il est préférable d'insérer des marqueurs courts fréquemment plutôt que des marqueurs longs moins fréquemment.

L'insertion de marqueurs binaires, plutôt que de symboles spécifiques codés arithmé-

tiquement permet un contrôle plus précis de la redondance ajoutée. De plus, aucun traitement particulier n'est requis pour conserver la mémoire de la source lors de l'insertion du marqueur.

3.8.2 Information adjacente

Comme nous l'avons signalé dans la section précédente, les marqueurs de synchronisation ne constituent pas des "observations parfaites", mais plutôt des repères qui favorisent la synchronisation de la séquence décodée. On peut considérer l'utilisation de points de synchronisation parfaitement fiables. Des valeurs du compteur de bit N_k ou du compteur de symboles K_n peuvent être transmises parallèlement au train binaire, puis utilisées respectivement par l'estimation à horloge symbole et l'estimation à horloge bit. L'utilisation de cette information adjacente va améliorer significativement les résultats, à condition qu'elle ne soit pas erronée. Par conséquent, il est nécessaire de la protéger efficacement contre les erreurs de transmission, en utilisant par exemple un code correcteur d'erreur. Le surcoût induit par cette protection limite l'utilisation de l'information adjacente, même si celle-ci est compressée.

3.8.3 Décodage turbo itératif

Les mécanismes de synchronisation décris précédemment permettent d'augmenter significativement la fiabilité de l'estimation de la séquence de symboles. On peut également considérer l'utilisation d'un code correcteur d'erreur. Nous proposons d'utiliser un code convolutif systématique (CC). Ce code peut être concaténé avec le code arithmétique, dans l'esprit des turbo codes en série. A partir de ce principe, on peut utiliser les deux modèles séparément dans un schéma itératif, à condition de les séparer par un entrelaceur [GFGR01]. La structure des dépendances entre les variables d'un tel schéma est représentée graphiquement sur la figure 3.7. A partir de la séquence de bits U_1^N, le codeur de canal dont les états sont notés X' génère la séquence de bits de redondance R_1^N. Les observations Z_1^N de ces bits sont disponibles à la sortie du canal. Le rendement du code de canal est controlé par poinçonnage.

Un tel schéma nécessite de transmettre l'information extrinsèque sur les bits U_n du décodeur de codes convolutifs vers le décodeur arithmétique souple, et réciproquement. Cette information représente la modification induite sur la loi conditionnelle de U_n sachant Y_n par l'introduction du reste des observations Y_1^{n-1}, Y_{n+1}^N. Cette information extrinsèque peut s'exprimer

$$Ext_{U_n}(Y|Y_n) \propto \frac{\mathbb{P}(U_n|Y)}{\mathbb{P}(U_n|Y_n)}. \tag{3.10}$$

L'estimation itérative consiste tout d'abord à appliquer un algorithme BCJR [BCJR74] (également annexe C) sur le code de canal. L'information extrinsèque sur les bits U_n est directement produite par cet algorithme. Elle sert d'entrée pour l'estimation effectuée sur le modèle de codeur ou de décodeur arithmétique. Le résultat de cette estimation est constitué des probabilités a posteriori $\mathbb{P}(X_K, N_K|Y)$ sur les derniers états (X_K, N_K) du

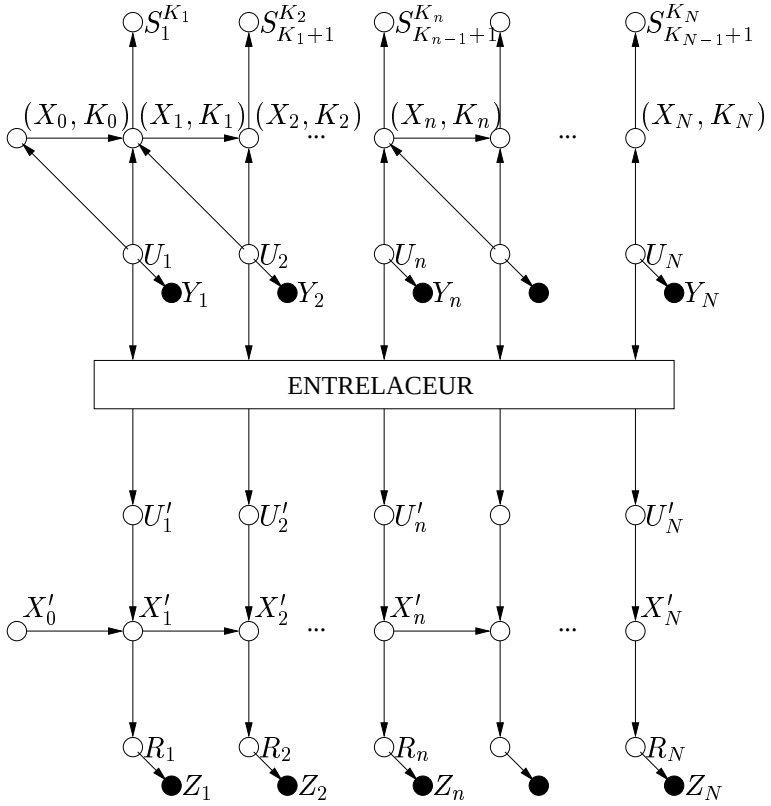


FIG. 3.7 – *Représentation graphique à horloge bit des dépendances dans la chaîne de codage conjoint arithmétique-codage de canal. Les points blancs et noirs sont respectivement les variables cachées et observées.*

modèle à horloge symbole ou $\mathbb{P}(X_N, K_N|Y)$ sur les derniers états (X_N, K_N) du modèle à horloge bit. Ces probabilités s'appliquent à des séquences de symboles entières et doivent être convertie en information sur les bits U_n par l'équation

$$P(U_n = i|Y)|_{i=0,1} = \sum_{(x_K, N_K):U_n=i} \mathbb{P}(x_K, N_K|Y), \tag{3.11}$$

à horloge symbole ou

$$P(U_n = i|Y)|_{i=0,1} = \sum_{(x_N, K_N):U_n=i} \mathbb{P}(x_N, K_N|Y), \tag{3.12}$$

à horloge bit.

3.8.4 Résultats expérimentaux

Pour évaluer les performances de la technique de décodage souple de codes arithmétiques, des séries d'expériences ont été réalisées sur des sources de Gauss-Markov de moyenne nulle, de variance unité et pour différents facteurs de corrélation ρ. La source est quantifiée uniformément sur 8 niveaux (3 bits) dans l'intervalle $[-3, 3]$. Toutes les simulations ont été effectuées pour un canal gaussien à bruit blanc additif (AWGN) avec une modulation BPSK. Les résultats sont moyennés au minimum sur 100 réalisations. Plusieurs mises en oeuvre de l'algorithme d'estimation sont possibles. Nous utilisons une stratégie "en largeur" d'exploration de l'arbre de recherche, c'est à dire que l'arbre de recherche est construit niveau par niveau ("breadth first strategy" dans [PHS01]).

La première série d'expérience permet de comparer les performances en termes de taux d'erreur symbole (SER) et de rapport signal à bruit (SNR) de l'algorithme de décodage souple de codes arithmétiques avec le décodage instantané de codes arithmétiques et le décodage souple de codes de Huffman [GFGR01]. Lorsque la corrélation de la source augmente, l'efficacité de compression du codage arithmétique devient nettement supérieure à celle du codage de Huffman. Des marqueurs de synchronisation sont alors ajoutés pour permettre une comparaison à débit égal des deux techniques de compression.

Les figures 3.8, 3.9 et 3.10 présentent les courbes de SER et SNR résiduels en fonction du rapport signal à bruit E_b/N_0 du canal, respectivement pour $\rho = 0.1$, $\rho = 0.5$ et $\rho = 0.9$. La première observation est que le décodage souple de codes arithmétique apporte un gain significatif par rapport au décodage instantané. Pour des sources faiblement corrélées ($\rho = 0.1$), le décodage souple de codes arithmétiques et de codes de Huffman produisent des résultats comparables, avec un léger avantage pour les codes de Huffman. Cela s'explique par le fait que pour des sources faiblement corrélées, le gain apporté par le codage arithmétique par rapport au codage de Huffman est limité. Par conséquent, peu de redondance peut être ajouté au code arithmétique. Les codes de Huffman ont alors une meilleure capacité de resynchronisation. En revanche, pour des sources plus corrélées, le décodage souple de codes arithmétiques surpasse le décodage souple de codes de Huffman. En effet, le gain en compression apporté par le codeur

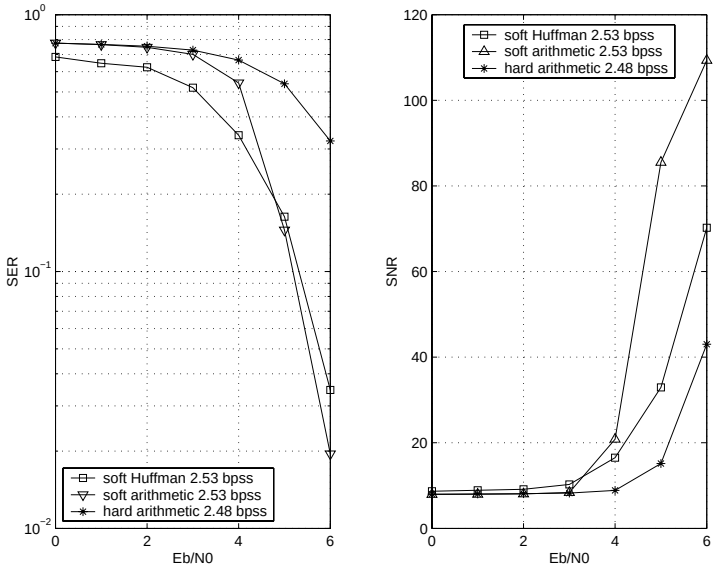


FIG. 3.8 – *SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman* ($\rho = 0.1$, $K = 200$ *symboles,* 100 *réalisations).*

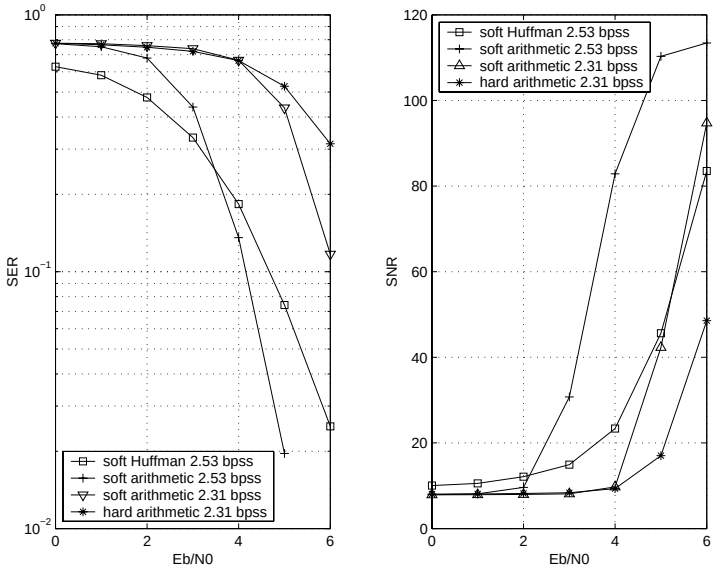


FIG. 3.9 – *SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman* ($\rho = 0.5$, $K = 200$ *symboles,* 100 *réalisations).*

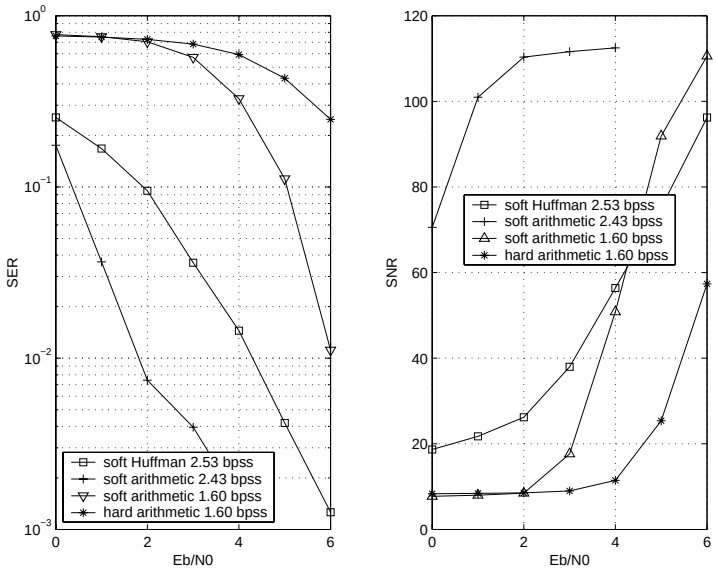


FIG. 3.10 – *SER et SNR pour (a) le décodage souple de codes arithmétiques, (b) le décodage arithmétique instantané et (c) le décodage souple de codes de Huffman* ($\rho = 0.9$, $K = 200$ *symboles,* 100 *réalisations).*

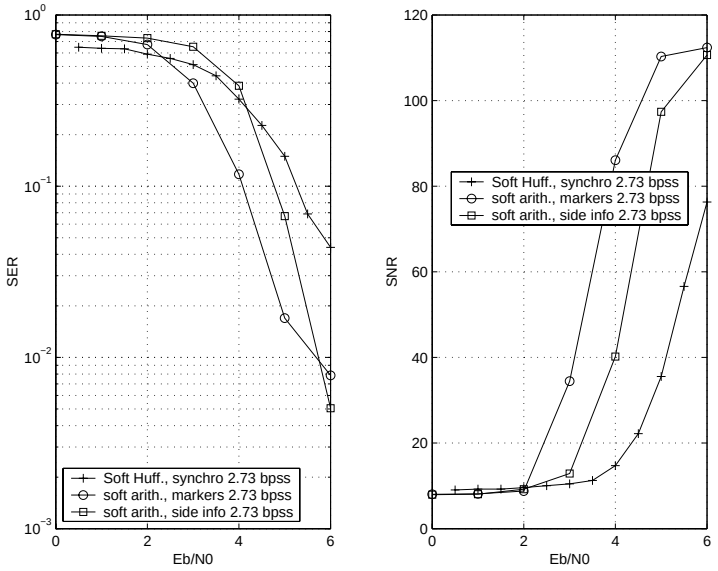


FIG. 3.11 – *SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation* ($\rho = 0.1$, $K = 200$ *symboles,* 100 *réalisations).*

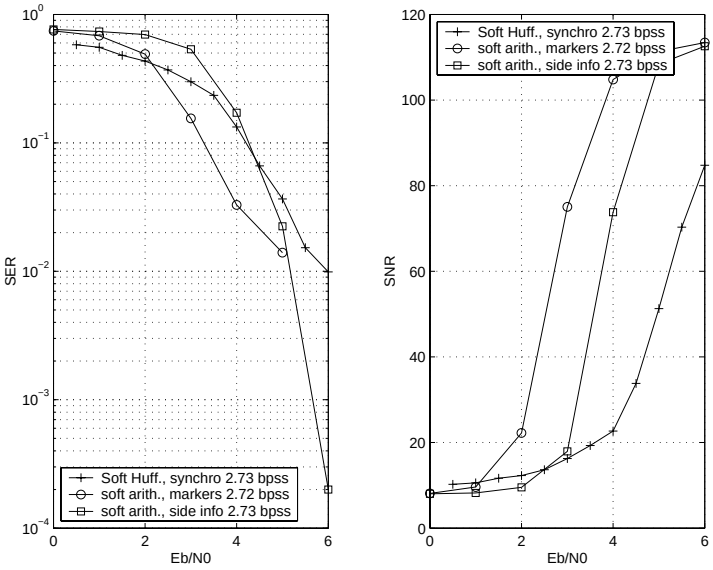


FIG. 3.12 – *SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation* ($\rho = 0.5$, $K = 200$ *symboles,* 100 *réalisations).*

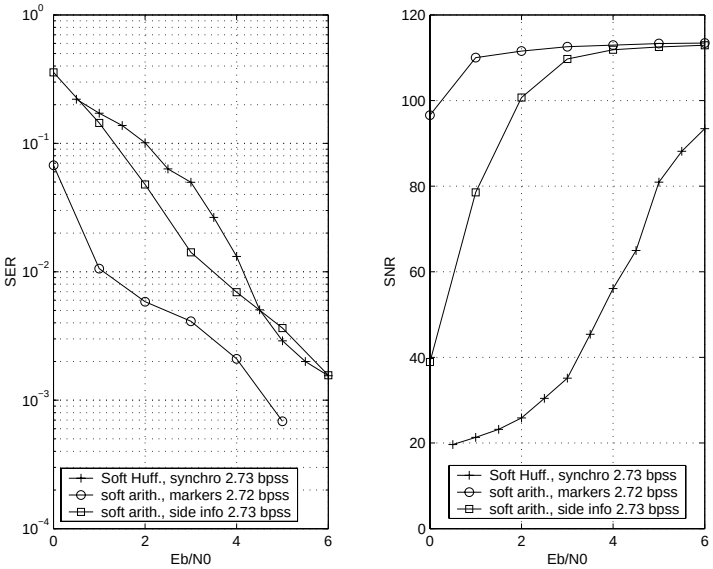


FIG. 3.13 – *SER et SNR pour (a) le décodage souple de codes arithmétiques avec des marqueurs de synchronisation, (b) avec de l'information adjacente et (c) le décodage souple de codes de Huffman avec des marqueurs de synchronisation* ($\rho = 0.9$, $K = 200$ *symboles,* 100 *réalisations).*

arithmétique permet d'ajouter une quantité importante de marqueurs de synchronisation, d'où un meilleur comportement face aux erreurs.

La deuxième série d'expériences permet d'évaluer les effets de l'ajout de redondance sous forme de marqueurs de synchronisation ou d'information adjacente. Dans ce dernier cas, le coût de protection de cette information a été pris en compte dans le calcul du débit. Les figures 3.11, 3.12 et 3.13 présentent les courbes de SER et SNR résiduels respectivement pour $\rho = 0.1$, $\rho = 0.5$ et $\rho = 0.9$. Encore une fois, le décodage souple de codes arithmétiques atteint de meilleures performances pour les sources fortement corrélées. Les facteurs de compression plus élevés obtenus avec le codage arithmétique apportent davantage de flexibilité pour l'ajout de redondance. Ainsi le problème de la synchronisation, qui est crucial pour les codes à longueur variables, est traité spécifiquement. Pour des sources moins corrélées ($\rho \leq 0.5$), le décodage souple de codes arithmétiques surpasse le décodage souple de codes de Huffman pour $E_b/N_0 \geq 2$ dB. On constate également que la synchronisation souple surpasse la transmission périodique d'information adjacente. Le coût des marqueurs de synchronisation étant inférieur, leur fréquence peut être plus importante pour un niveau de redondance équivalent, ce qui permet une resynchronisation plus fréquente du décodeur. Cependant, l'utilisation de l'information adjacente est beaucoup plus efficace en terme d'élagage. L'estimation est alors entre 2 et 10 fois plus rapide.

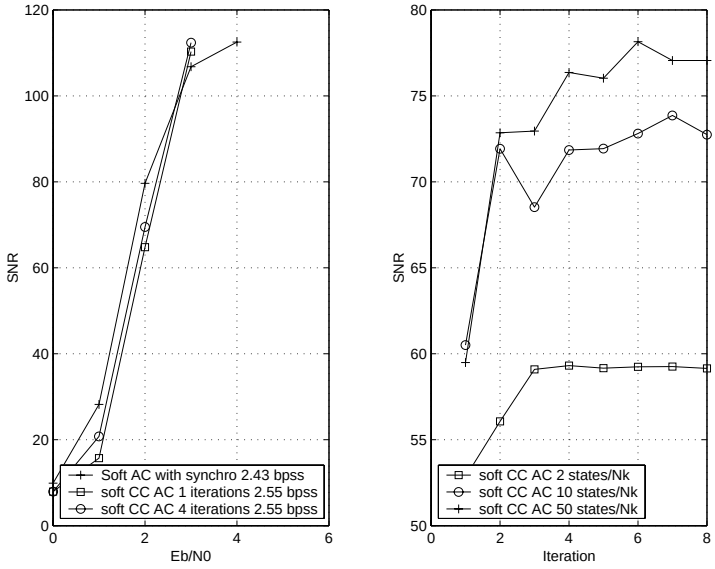


FIG. 3.14 – *SNR pour le décodage souple itératif de codes arithmétiques ($\rho = 0.1$, 100 réalisations) (a) comparé à la synchronisation souple ($K = 200$), (b) en fonction du nombre d'itérations pour différents paramètres d'élagage ($w = 2, 10, 50$, $K = 20$).*

La figure 3.14-a présente les performances du schéma de décodage souple itératif codes convolutif-codes arithmétiques (CC-AC) en fonction de E_b/N_0. Bien que les résultats observés soient bon, on remarque que les itérations n'apportent pas de gain si-

gnificatif. Cette contre-performance provient de l'élagage qui est utilisé lors du décodage souple de codes arithmétiques, comme cela est mis en évidence par la figure 3.14-b. Le décodeur arithmétique transmet une information extrinsèque qui ne concerne qu'un sous-ensemble de toutes les solutions possibles au décodeur de canal, ce qui limite l'effet de ce dernier. A l'itération suivante, le décodeur arithmétique ne disposera d'information que sur les solutions qu'il a déjà trouvé les plus vraisemblables précédemment. Les itérations n'apportent alors pas ou peu d'innovation.

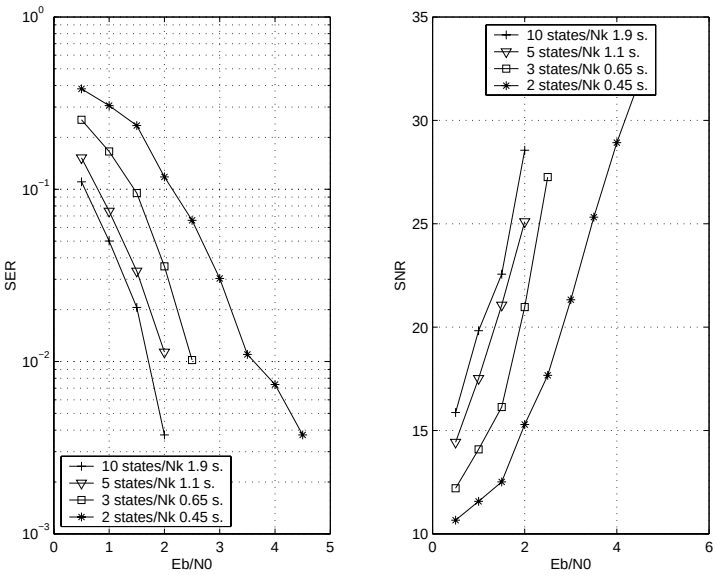


FIG. 3.15 – *Illustration du compromis entre performance et complexité* ($w = 2, 3, 5, 10$).

Enfin pour conclure, la figure 3.15 présente un résultat indicatif sur le rapport entre la complexité et la robustesse de l'algorithme de décodage souple de codes arithmétiques en fonction du paramètre d'élagage.

3.9 Enrichissement de JPEG2000 par décodage souple de codes arithmétiques

Les performances en compression d'un codeur arithmétique dépendent de la précision du modèle de source utilisé. Des gains substantiels peuvent être obtenus par l'utilisation de modèles plus sophistiqués. Ces modèles peuvent être adaptatifs, comme par exemple dans H.264 et MPEG-4 part 10, ou bien être basés sur des contextes dépendants d'un ou plusieurs symboles précédents, comme par exemple dans JPEG2000. Dans ce dernier cas, la source ne peut plus être considérée comme une chaîne de Markov d'ordre 1. Cependant, les algorithmes développés ci-dessus sont toujours applicables, comme nous le montrons à présent dans le cas de JPEG2000.

3.9.1 Codage arithmétique contextuel dans EBCOT

L'élément clé de JPEG2000 est l'algorithme EBCOT (Embedded Block Coding with Optimized Truncation), qui découpe les sous-bandes du domaine ondelettes en blocs, code ces blocs puis génère le train binaire. Le codage arithmétique intervient au niveau du codage de ces blocs, appelés code-blocks. Un code-block peut être vu comme un tableau d'entiers en représentation signe-amplitude. Il est codé plan de bit par plan de bit par un codeur arithmétique contextuel. Comme cela est décrit en annexe A, et rappelé en section 2.1, les plans de bits sont divisés en trois sous-plans de bits, ou passes, de façon à obtenir une progressivité plus fine. Ainsi chaque plan de bit est parcouru trois fois. Suivant l'information contenue dans chaque bit à coder, quatre primitives binaires sont utilisées pour les représenter :

- La primitive ZC (Zero Coding) représente les bits des emplacements qui n'ont pas encore été trouvés signifiants. Cette primitive indique quand un emplacement devient signifiant.
- La primitive RLC (Run Length Coding) représente un ensemble de primitives ZC de mêmes valeurs. Elle apporte un léger gain en compression.
- La primitive SC (Sign Coding) représente le signe d'un entier. Elle est utilisée dès qu'un emplacement est trouvé signifiant.
- La primitive MR (Magnitude refinement) représente les bits des emplacements qui ont déjà été trouvés signifiants. Elle permet de raffiner l'information déjà connue pour un emplacement.

La première passe de codage est appelée "passe de propagation des signifiants". Un bit est codé durant cette passe si son emplacement n'a pas encore été trouvé signifiant et si au moins un de ses voisins a déjà été trouvé signifiant. Les primitives ZC et RLC sont utilisées durant cette passe, ainsi que la primitive SC chaque fois qu'un emplacement devient signifiant (une primitive ZC égale à 1 est immédiatement suivie d'une primitive SC). La deuxième passe de codage est appelée "passe de raffinement d'amplitudes". Durant cette passe, les bits des emplacements qui ont déjà été trouvés signifiants dans les plans de bits précédents sont codés avec la primitive MR. Finalement, la troisième passe de codage est appelée "passe de nettoyage". Elle visite tous les emplacements restants et utilise les primitive ZC, RLC et SC si nécessaire pour les coder.

En résumé, durant chaque passe de codage, la décision d'inclure ou non un bit dans cette passe dépend du *contexte* de ce bit. Le contexte d'un bit est défini à partir de la signifiance de son emplacement et de la signifiance et du signe des emplacements voisins. Quand un bit est inclu dans une passe, il est représenté par une des quatre primitives binaires, puis codé arithmétiquement avec le *MQ-coder* [ISO00]. Le modèle statistique utilisé par ce codeur est contextuel. Ce modèle est pre-défini dans le standard JPEG2000 pour tous les contextes possibles. Ainsi les contextes définissent non seulement les probabilités utilisées par le codeur arithmétique, mais aussi l'ordre de parcours des bits de chaque plan de bits.

3.9.2 Application de l'algorithme de décodage souple

Jusqu'à présent, dans la présentation des modèles et algorithmes pour le décodage souple de codes arithmétiques, nous avons considéré des sources de Markov d'ordre 1. Le codeur et le décodeur exploitent tous les deux les probabilités stationnaires $P(s_i)$ et de transition $P(s_i|s_{i-1})$, respectivement pour approcher au mieux l'entropie et pour obtenir l'estimation la plus fiable. Dans le cas du codage arithmétique contextuel dans JPEG2000, une source binaire dont la distribution dépend du contexte associé à chaque bit est considérée. Nous notons S_k le k-ième bit visité dans un plan de bit et C_k le contexte associé à ce bit. Du fait de la technique de codage du plan de bit en trois passes, *l'ordre de balayage indexé par k dépend du contexte*. Les probabilités $P(S_k|C_k)$ sont spécifiées dans le standard JPEG2000.

L'estimation à horloge symbole repose sur l'équation 3.2, où le modèle de source est exploité par l'expression de $P(S_1^k)$ en fonction de $P(S_1^{k-1})$ et de la probabilité de transition de S_1^{k-1} vers S_k. Dans le cas du modèle contextuel, on a :

$$P(S_1^k) = P(S_1^{k-1}) \cdot P(S_k|C_k), \tag{3.13}$$

et l'algorithme d'estimation reste valide. De même, si on considère l'estimation à horloge bit, la dépendance aux contextes est introduite dans l'équation 3.8 par l'intermédiaire de

$$\mathbb{P}(S_{K_{n-1}+1}^{K_n}|S_1^{K_{n-1}}) = \begin{cases} \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k|C_k) \textbf{ si } K_n > K_{n-1} \\ 1 \textbf{ sinon.} \end{cases} \tag{3.14}$$

Quel que soit le modèle utilisé, les états (X_k, N_k) ou (X_n, K_n) qui sont estimés doivent contenir non seulement l'état du codeur arithmétique, mais aussi l'information concernant les contextes, qui permet de déterminer à la fois les probabilités des bits et l'ordre de parcours des plans de bits. Même si la taille des code-blocks est connue, le nombre de primitives binaires utilisées pour coder un code-block varie. Le nombre de symboles à décoder n'est pas connu par le décodeur. Il est donc préférable d'effectuer l'estimation à horloge bit.

3.9.3 Amélioration des performances par ajout de redondance

Comme précédemment, l'efficacité du décodage souple de codes arithmétiques peut être renforcée en ajoutant de la redondance de manière appropriée. On remarque tout d'abord que le fait que l'algorithme EBCOT sépare les sous-bandes en blocs codés indépendamment favorise l'utilisation du décodage souple de codes arithmétiques. En effet, nous avons vu que la complexité de notre algorithme était exponentielle, d'où l'utilisation de techniques d'élagage. De ce fait, il est plus intéressant d'appliquer cet algorithme sur plusieurs trains binaires indépendants plutôt que sur un seul train binaire plus long, contenant la totalité de l'information. De plus, le découpage en blocs

limite la propagation d'erreur spatialement dans les sous-bandes. Ainsi, on pourra faire varier les performances du système en modifiant la taille des code-blocks. En revanche, le découpage en blocs et l'arrangement du train binaire qui est effectué par la suite nécessite la transmission d'information permettant de segmenter correctement ce train binaire. Cette information est présente dans les entêtes des *paquets EBCOT*. Ne pouvant tolérer la moindre erreur sur ces entêtes, nous les avons protégés à l'aide d'un code convolutif de rendement élevé.

Pour ce qui concerne les données images à proprement parler, nous avons adopté un mécanisme de synchronisation souple, comme décrit dans la section 3.8.1. Cette stratégie est particulièrement adaptée, puisque le standard JPEG2000 inclut la possibilité d'insérer des marqueurs sous la forme de symboles spécifiques codés arithmétiquement. Ces marqueurs sont insérés à la fin de chaque plan de bit et sont utilisés pour de la détection d'erreur. Nous avons modifié le codeur JPEG2000 de façon à pouvoir insérer ces marqueurs soit à la fin de chaque plan de bits, soit à la fin de chaque passe, soit à la fin de chaque *stripe* (les code-blocks sont parcourus par bandes horizontales appelées "stripe"). L'algorithme d'estimation utilise ces marqueurs comme marqueurs de resynchronisation.

3.9.4 Résultats expérimentaux

L'algorithme d'estimation a été introduit dans JPEG2000 et une série d'expériences a été effectuée. Les marqueurs de détection d'erreurs spécifiés par le standard ont été systématiquement activés, en plus des marqueurs de synchronisation insérés à la fin de chaque "stripe". Tous les entêtes et marqueurs de syntaxe ont été protégés par un code de canal convolutif systématique de rendement 1/3, défini par les polynômes $F(z) = 1 + z + z^2 + z^4$ et $G(z) = 1 + z^3 + z^4$.

Les figures 3.16 et 3.17 présentent les PSNR obtenus pour Lena 512×512 compressée respectivement à 0.25 et 0.5 bits par pixel en fonction du rapport signal à bruit E_b/N_0 du canal. Le décodage instantané avec le standard JPEG2000 présente une faible résistance aux erreurs. Les algorithmes d'estimation proposés apportent un gain significatif. La figure 3.18 permet d'évaluer la qualité visuelle des résultats obtenus sur l'image Lena 512×512 compressée à 0.5 bits par pixel, pour un canal AWGN avec $E_b/N_0 = 5db$. Avec le décodage instantané, la propagation des erreurs entraîne une perte totale de l'information (figure 3.18(c)). L'algorithme d'estimation permet de récupérer une image de qualité satisfaisante, même si des artefacts peuvent subsister.

3.10 Conclusion

Les réseaux bayésiens constituent un cadre adapté à l'analyse des structures de dépendances entre les variables impliquées dans un processus de codage et de décodage. La conception d'algorithmes d'estimation s'ensuit naturellement. Dans le cas des codes arithmétiques, la première difficulté provient de l'absence de correspondance un à un entre les symboles codés et les mots de codes binaires. Un modèle d'états du codeur permet d'associer les bits observés aux symboles de la séquence. La seconde difficulté

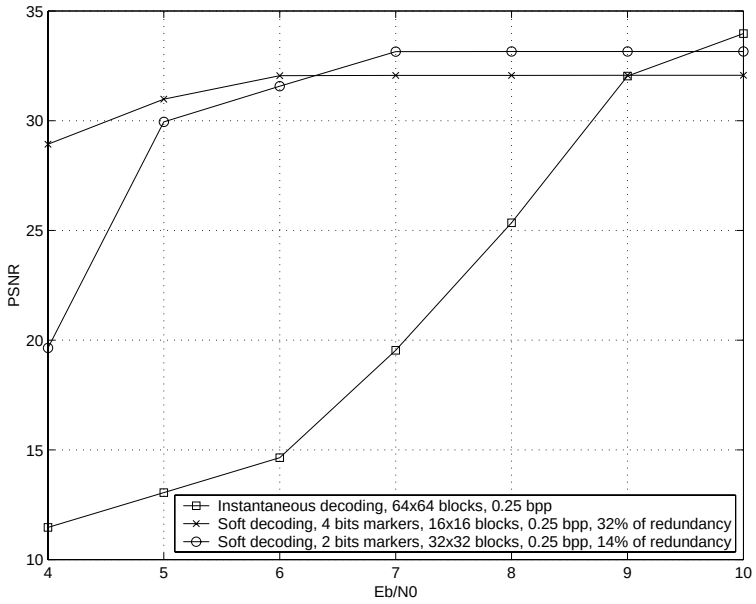


FIG. 3.16 – *Décodage souple de Lena 512x512 codée par JPEG2000 à 0.25 bits par pixel.*

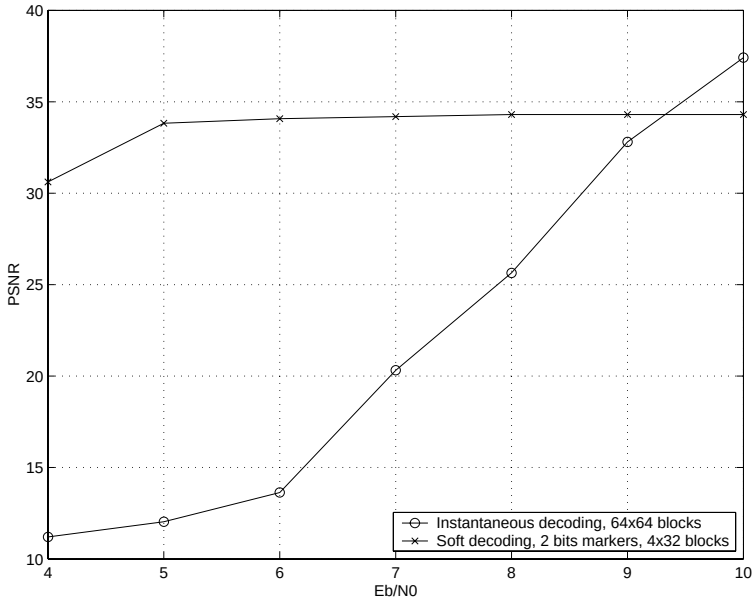
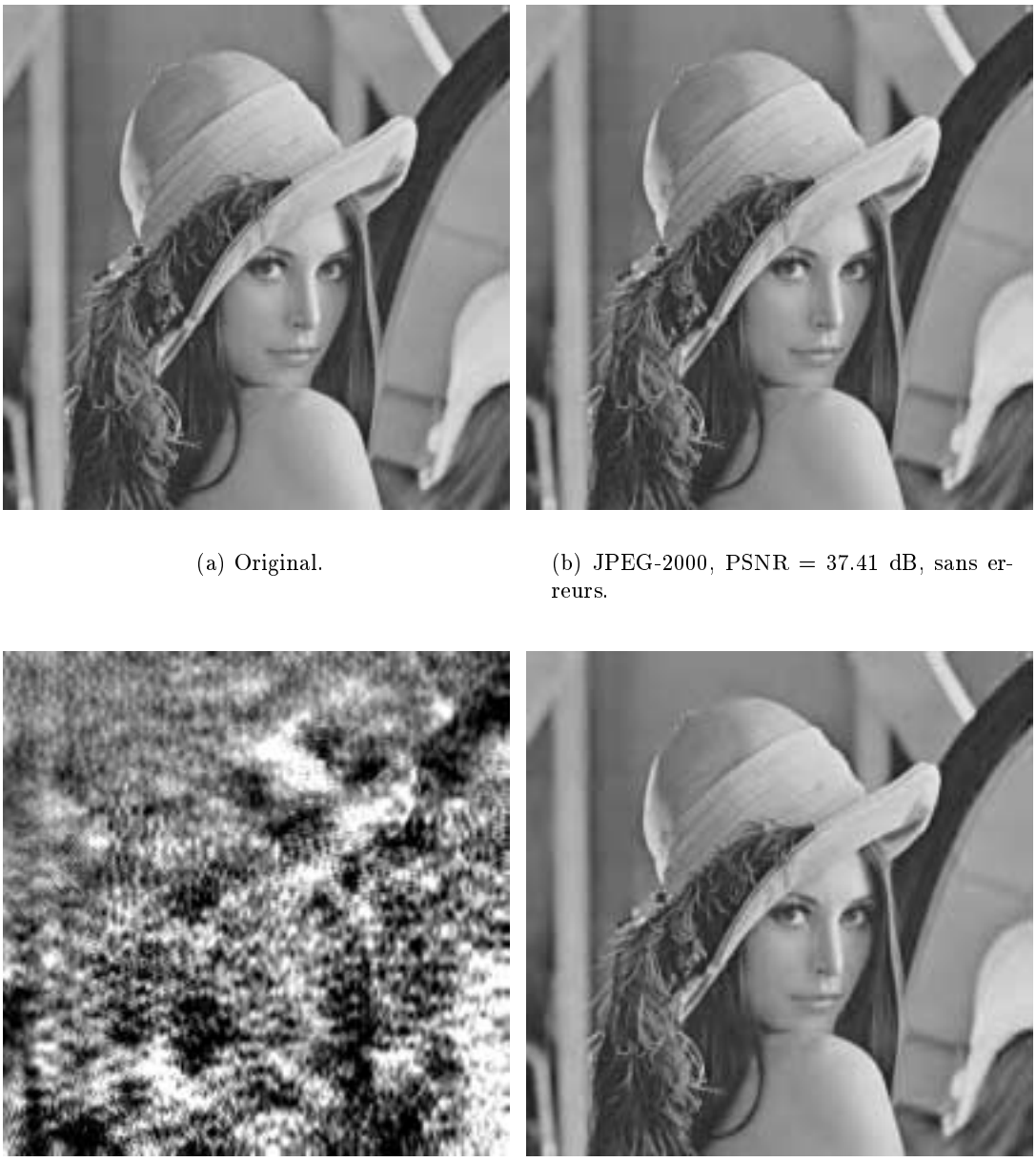


FIG. 3.17 – *Décodage souple de Lena 512x512 codée par JPEG2000 à 0.5 bits par pixel.*



(a) Original.

(b) JPEG-2000, PSNR = 37.41 dB, sans erreurs.

(c) JPEG-2000, canal AWGN ($E_b/N_0 = 5$ dB), PSNR = 11.25 dB.

(d) JPEG-2000 et décodage souple, canal AWGN ($E_b/N_0 = 5$ dB), PSNR = 33.26 dB.

FIG. 3.18 – *Impact des erreurs sur la qualité visuelle (0.5 bpp).*

provient de la complexité qui croît exponentiellement avec le nombre de symboles codés. Une stratégie d'élagage adaptée permet de circonvenir cette difficulté, au prix d'une perte de fiabilité de l'estimation. Les algorithmes d'estimation ont tout d'abord été développés pour des sources de Markov d'ordre 1, mais ils s'adaptent aisément à des modèles d'ordre supérieurs. Ainsi, nous les avons adaptés au codage arithmétique contextuel utilisé dans le standard JPEG2000. Les résultats expérimentaux sur des images montrent la validité de l'approche.

L'exploitation de redondance supplémentaire, sous la forme de marqueurs de synchronisation souple, d'information adjacente ou de codage de canal permet d'augmenter l'efficacité de l'algorithme proposé. Le compromis compression/redondance est alors aisément adapté aux conditions de transmissions. Cette redondance supplémentaire permet non seulement de favoriser la fiabilité de l'estimation, mais aussi de limiter la croissance du treillis en favorisant la stratégie d'élagage employée. Le code de canal peut être placé dans un schéma itératif inspiré des turbo codes en série. Cependant, le gain apporté par les itérations est limité par la sous-optimalité de l'algorithme d'estimation, inhérente à la stratégie d'élagage. Pour pouvoir définir un algorithme d'estimation optimal, il est nécessaire de revoir le principe du codage arithmétique. Dans le chapitre suivant, nous étudions l'utilisation de codes arithmétiques simplifiés, dits *codes quasi-arithmétiques*.

Chapitre 4

Décodage souple de codes quasi-arithmétiques

4.1 Introduction

Les systèmes de compression utilisés en codage d'audio, d'image et de vidéo incluent généralement une étape de codage entropique, et plus particulièrement, pour les systèmes récents, de codage arithmétique. Dans le cadre de la transmission de données sur des canaux à erreurs bits, lorsque la capacité de correction du codage de canal est dépassée, les données décodées sont entâchées d'erreurs bits résiduelles. Dans ce contexte, le codage arithmétique devient un point faible. En effet, une seule erreur bit suffit à désynchroniser le décodeur. Tous les symboles suivant l'erreur sont alors erronés.

Ce constat a justifié, dans le chapitre précédent, le développement d'un algorithme de décodage souple de codes arithmétiques. La structure des dépendances entre les variables impliquées dans le processus de codage et de décodage a été modélisée par un réseau bayésien (voir aussi annexe B). Cela a permis non seulement de déduire l'algorithme de décodage souple, mais aussi de permettre à cet algorithme d'exploiter différentes façons d'ajouter de la redondance et ainsi d'augmenter la robustesse du schéma proposé. Cet algorithme a été validé expérimentalement sur des sources théoriques et dans un codeur d'images JPEG2000. Il s'est révélé particulièrement efficace, montrant ainsi l'intérêt de traiter spécifiquement le problème de la désynchronisation des codes arithmétiques lors de transmissions bruitées.

Cependant, la structure des codes arithmétiques entraîne une croissance exponentielle du nombre d'états du treillis de codage ou de décodage. Par conséquent, il a été nécessaire de définir une stratégie d'élagage qui rend l'algorithme de décodage souple sous-optimal. L'impact de l'élagage sur les performances de l'algorithme, bien que modéré, a pu être mis en évidence dans le cas d'un schéma de décodage itératif inspiré des turbo codes en série, où un codeur convolutif est concaténé au codeur arithmétique. Ce défaut de l'algorithme de décodage souple provient uniquement de la structure des codes arithmétiques. Si l'on souhaite concevoir un schéma d'estimation optimal, il convient de remettre en cause le principe du codage arithmétique. Nous avons vu dans

le chapitre précédent que pour des canaux fortement bruités, il est nécessaire d'ajouter de la redondance dans les données compressées pour atteindre une robustesse suffisante. Si l'on admet le principe d'ajouter de la redondance, on peut admettre le fait que le codeur entropique ne soit pas optimal. Une première solution pour avoir un algorithme d'estimation optimal consiste à revenir aux codes de huffman et à utiliser la méthode de décodage souple proposée dans [GFGR01]. Cependant, les codes de Huffman sont moins flexibles que les codes arithmétiques. En codage arithmétique, le découplage du codeur et du modèle de source permet d'utiliser des modèles de sources complexes, voire adaptatifs. De plus, les codes de Huffman ne permettent pas de compresser des sources binaires.

Nous proposons donc un nouveau schéma de codage basé sur l'utilisation de codes arithmétiques simplifiés appelés *codes quasi-arithmétiques*. Suivant la précision adoptée, les performances en compression d'un codeur quasi-arithmétique varient entre celles d'un codeur de Huffman et celles d'un codeur arithmétique. Les codes quasi-arithmétiques ont le double avantage d'être aussi flexibles que les codes arithmétiques et de pouvoir, lorsque la précision est suffisament réduite, être modélisés par un automate à nombre d'états fini. Il est alors possible de s'affranchir du problème de la croissance exponentielle du nombre d'états rencontré dans le chapitre précédent.

Dans ce chapitre, nous nous appuyons de nouveau sur les réseaux bayésiens afin de formaliser les dépendances et les contraintes entre les variables d'une chaîne de codage composée d'une source de Markov, d'une conversion de source M-aire vers binaire et d'un codeur quasi-arithmétique. La conversion de source a pour but d'obtenir un meilleur compromis entre les performances en compression et le nombre d'états du codeur. L'ensemble de la chaîne peut être modélisé par un automate à nombre d'états finis. On cherchera à réduire au maximum ce nombre d'états. L'algorithme d'estimation se déduit naturellement et est optimal. Il permet la prise en compte de redondance sous forme de marqueurs de synchronisation, d'information adjacente ou bien de codes de canal. Dans ce dernier cas, il est possible de concevoir un schéma itératif efficace. Les résultats expérimentaux permettent de valider l'apport de ce nouveau schéma de codage robuste par rapport à celui présenté dans le chapitre précédent.

4.2 Notations

Les notations diffèrent légèrement de celles employées au chapitre précédent. Soit $A = A_1 \dots A_L$ une séquence de symboles de source quantifiés qui prennent leur valeur dans un alphabet fini $\mathcal{A}$ composé de $M = 2^q$ symboles, $\mathcal{A} = \{a_1, a_2, \dots, a_i, \dots, a_M\}$. Nous supposons que la séquence $A = A_1 \dots A_L$ est une chaîne de Markov d'ordre 1. Si besoin, cette séquence de symboles M-aires peut être convertie en une séquence de symboles binaires $S = S_1 \dots S_K$, avec $K = q \times L$. Le source M-aire ou binaire est convertie à son tour en une séquence de bits d'information $U = U_1 \dots U_N$, par l'intermédiaire d'un codeur quasi-arithmétique de précision T. La longueur N de ce train binaire est une variable aléatoire, fonction de A. Le train binaire U est transmis sur un canal sans mémoire et reçu comme une mesure Y. Le problème que nous considérons

consiste à estimer A, connaissant les observations y. Nous utilisons les lettres majuscules pour dénoter les variables aléatoires, et les minuscules pour les réalisations de ces variables. Pour désigner une suite de variables successives, nous utilisons la notation $X_u^v = \{X_u, X_{u+1}, \ldots, X_v\}$ ou $\bar{X}_I$ avec I l'ensemble des index $\{u, u+1, \ldots, v\}$.

4.3 Codage arithmétique à précision réduite

Afin de définir des procédés de décodage robuste de codes arithmétiques, le processus de codage arithmétique peut se modéliser sous la forme d'un automate stochastique dont les états sont définis par trois variables : low, up correspondant respectivement aux bornes inférieures et supérieures de l'intervalle courant et n_{scl} recensant le nombre de mises à l'échelle effectuées depuis le dernier bit émis, comme expliqué dans la section 3.4.1. Les subdivisions de l'intervalle courant $[low, up[$, c'est à dire les états suivants, sont fonction des probabilités cumulées de la source. Si le modèle de source n'est pas connu, il existe une infinité de subdivisions possibles, d'où un nombre d'états infini. Si la distribution de la source est connue, le nombre d'états est fini, mais il croît exponentiellement avec le nombre de symboles codés. L'utilisation du codage arithmétique à précision réduite nous permet d'obtenir une modélisation par un automate stochastique à nombre d'états fini.

4.3.1 Codage quasi-arithmétique

Les auteurs de [HV92b] montrent que en réduisant la précision d'un codeur arithmétique, on peut réduire le nombre d'états possibles sans perte excessive en efficacité de compression. Dans ce cas, toutes les transitions entre états et les émissions de bits correspondantes peuvent être pré-calculées. Les opérations arithmétiques sont alors remplacées par des accès dans des tableaux (LUT). Cette mise en œuvre rapide et à précision réduite du codage arithmétique est appelée *codage quasi-arithmétique* [HV92b]. Pour effectuer le codage quasi-arithmétique, l'intervalle réel $[0, 1[$ est remplacé par l'intervalle entier $[0, T[$. Le paramètre T permet de contrôler le compromis entre la complexité et l'efficacité de compression : si T est grand, alors les subdivisions de l'intervalle représenteront précisément la distribution de la source. Par contre, si T est petit, toutes les subdivisions possibles de l'intervalle pourront être pré-calculées.

Une séquence A_1^L de symboles prenant valeur dans l'alphabet $\mathcal{A}$ de taille M est convertie en une séquence de bits U_1^N par un arbre M-aire. Cet arbre peut être vu comme un automate qui modélise la distribution du train binaire. Le codage d'un symbole détermine le choix d'un arc de l'arbre, auquel peut être associée l'émission d'une séquence de bits de longueur variable. Chaque nœud correspond à un état X du codeur arithmétique. Les transitions successives entre ces états suivent la distribution de la source ($\mathbb{P}(A_l|A_{l-1})$ pour une source de Markov d'ordre 1). Soit X_k l'état de l'automate à chaque *instant symbole* l. Comme pour le codage arithmétique, l'état X_l du codeur quasi-arithmétique est défini par trois variables : $lowA_l$, upA_l et $nscl_l$. Les termes $lowA_l$ et upA_l désignent les bornes de l'intervalle résultant des subdivisions successives de $[0, T[$ correspondant au codage de la séquence A_1^l. La quantité $nscl_l$ est

remise à zéro chaque fois qu'un bit est émis et incrémentée chaque fois qu'une remise à l'échelle sans émission de bit est effectuée. Elle représente donc le nombre de remises à l'échelle effectuées depuis la dernière émission de bit. Quand un bit est émis, il est suivi par $nscl_l$ bits de valeur opposée (section 3.3).

Dans la mesure où il y a un nombre fini de subdivisions possibles de l'intervalle $[0, T[$, il doit être possible de pré-calculer tous les états du codeur quasi-arithmétique sans connaître la source a priori. Il faut cependant tenir compte de la variable $nscl_l$ qui elle n'est pas bornée. La solution consiste alors à considérer $nscl_l$ comme une variable résultant des transitions entre états, et non plus comme appartenant aux états X_l pré-calculés. Le tableau 4.1 présente les états, les sorties et toutes les transitions possibles d'un codeur quasi-arithmétique de précision $T = 4$ pour une source binaire. La valeur de la variable $nscl_l$ n'est pas inclue dans le modèle d'état. Seuls les incréments de cette variable sont signalés par la lettre f dans le tableau, en référence au terme *follow up* utilisé dans [HV93]. Le codeur a trois états, correspondant aux subdivisions entières de l'intervalle $[0, 4[$. Les subdivisions suivantes de chacun de ces états sont fonction de la distribution de la source. Elles sont choisies de manière à ce que l'approximation correspondante de la distribution de la source minimise le débit excédent [HV93]. Considérons par exemple que l'automate est dans l'état $X_l = 1$, défini par l'intervalle $[lowA_l, upA_l[= [0, 3[$. Suivant la probabilité du symbole binaire d'entrée 0, l'intervalle $[0, 3[$ sera subdivisé en $[0, 2[$, correspondant à une probabilité approchée de $\frac{2}{3}$, si sa probabilité est supérieure à $\frac{1}{2}$, ou en $[0, 1[$, correspondant à une probabilité approchée de $\frac{1}{3}$, si sa probabilité est inférieure à $\frac{1}{2}$. Les deux subdivisions possibles conduisent, après avoir effectué les émissions de bits et les remises à l'échelle appropriées, à l'état 0. Le nombre d'états X_l possibles est $\frac{3T^2}{16}$. En tenant compte des différentes distributions de la source, le nombre de transitions possibles entre tous les états X_l est $\frac{9T^3}{64} - \frac{6T^2}{32} + \frac{T}{4}$.

				normal state model				simplified state model			
state	$[lowA_l,$	$\mathbb{P}(0)$, corresponding		0		1		MPS		LPS	
X_l	$upA_l[$	interval sub-division		out	next	out	next	out	next	out	next
0		$0.63 \leq \mathbb{P}(0)$	[0,3[	-	1	11	0	-	1	11	0
start	[0,4[	$0.37 \leq \mathbb{P}(0) < 0.63$	[0,2[	0	0	1	0	0	0	1	0
		$\mathbb{P}(0) < 0.37$	[0,1[	00	0	-	2				
1	[0,3[	$0.5 \leq \mathbb{P}(0)$	[0,2[	0	0	10	0	0	0	10	0
		$\mathbb{P}(0) < 0.5$	[0,1[	00	0	f	0				
2	[1,4[	$0.5 \leq \mathbb{P}(0)$	[1,3[	f	0	11	0				
		$\mathbb{P}(0) < 0.5$	[1,2[	01	0	1	0				

TAB. 4.1 – *Etats, transitions et sorties d'un codeur quasi-arithmétique avec* $T = 4$.

Le nombre d'états peut être encore réduit en identifiant les symboles binaires comme *moins probable (LPS)* et *plus probable (MPS)*, plutôt que 0 et 1. Cela revient à réduire le nombre de combinaisons possibles des probabilités de la source binaire, les MPS et LPS pouvant être égaux soit à 0 soit à 1. Cela permet de combiner certaines transitions et d'éliminer des états, comme cela est montré sur la partie droite du tableau 4.1. La

séquence $X_0 \dots X_L$ d'états successifs du codeur quasi-arithmétique forme une chaîne de Markov et les sorties du codeur sont fonctions des transitions sur cette chaîne. Cette modélisation permet l'élaboration de décodeurs robustes optimaux au sens du MAP.

4.3.2 Décodage quasi-arithmétique

De la même manière que le codage quasi-arithmétique, le décodage peut se modéliser par un automate stochastique à nombre d'états fini. Considérons tout d'abord un codeur arithmétique classique : une séquence U_1^n de bits codés arithmétiquement est convertie en une séquence de symboles $A_1^{L_n}$ par un arbre binaire. Chaque bit détermine le choix d'un arc dans l'arbre. A chaque nœud ν de cet arbre correspond une séquence U_1^{n-1}, à partir de laquelle deux transitions $U_n = 0$ ou $U_n = 1$ sont possibles. Ces nœuds correspondent aux états du décodeur arithmétique. L'état d'un décodeur arithmétique est défini par deux intervalles : $[lowU_n, upU_n[$ et $[lowA_{L_n}, upA_{L_n}[$. L'intervalle $[lowU_n, upU_n[$ correspond au segment de l'intervalle $[0, 1[$ défini par la réception de la séquence de bits U_1^n. L'intervalle $[lowA_{L_n}, upA_{L_n}[$ est le segment de l'intervalle $[0, 1[$ obtenu quand la séquence de symboles $A_1^{L_n}$ a été décodée. Cet intervalle est mis à jour de la même manière qu'au codage à chaque symbole décodé. Ces deux intervalles doivent être mis à l'échelle de manière appropriée afin d'éviter les problèmes de précision numérique. Cette remise à l'échelle s'effectue de la même manière qu'au codage, à partir des symboles décodés. Cependant, il est inutile de conserver la trace des remises à l'échelle qui ont été effectuées.

Considérons à présent le processus de décodage sur l'intervalle entier $[0, T[$. Le décodeur quasi-arithmétique peut également être exprimé sous la forme d'un automate stochastique. Soit X_n l'état de cet automate à *l'instant bit* n. X_n contient les quatre variables $[lowU_n, upU_n[$ et $[lowA_{L_n}, upA_{L_n}[$. Comme il y a un nombre fini de subdivisions possibles de l'intervalle $[0, T[$, l'automate qui définit le décodeur quasi-arithmétique a un nombre fini d'états, qui peuvent être pré-calculés. Le tableau 4.2 présente les états, transitions et sorties d'un décodeur quasi-arithmétique pour une source binaire, avec $T = 4$ et la simplification MPS/LPS. Dans ce cas particulier, le décodeur a seulement deux états. Les subdivisions des intervalles correspondant à ces deux états dépendent de la distribution de la source (désignée par $\mathbb{P}(MPS)$ dans le tableau 4.2). Considérons par exemple que l'automate est dans l'état $X_n = 0$, défini par les deux intervalles $[lowU_n, upU_n[= [0, 4[$ et $[lowA_{L_n}, upA_{L_n}[= [0, 4[$, et que le bit d'entrée est $U_n = 0$. Suivant les probabilités de la source codée (dans ce cas $\mathbb{P}(MPS)$ et $\mathbb{P}(LPS)$), l'intervalle $[lowA_{L_n}, upA_{L_n}[= [0, 4[$ sera subdivisé en $[0, 3[$ (si la probabilité du MPS est supérieure à 0.63) ou en $[0, 2[$ (si la probabilité du MPS est inférieure à 0.63). Dans les deux cas, la transition correspondante retourne à l'état $X_n = 0$.

4.3.3 Approximation de la distribution de la source

Le codage arithmétique peut être vu comme une conversion d'une distribution de probabilités vers une autre [Say99]. Un codeur arithmétique effectue une conversion d'une séquence de symboles M-aire de distribution donnée vers une séquence de

state X_n	state variables	$\mathbb{P}(MPS)$ (corresponding subdivision of $[lowA_{L_n}, upA_{L_n}[$)		$U_n = 0$ out	$U_n = 0$ next	$U_n = 1$ out	$U_n = 1$ next
0 (start)	$[lowU_n, upU_n[: [0, 4[$ $[lowA_{L_n}, upA_{L_n}[: [0, 4[$	$0.63 \leq \mathbb{P}(0)$ $0.37 \leq \mathbb{P}(0) < 0.63$	$([0, 3[)$ $([0, 2[)$	MPS, MPS MPS	0 0	- LPS	1 0
1	$[lowU_n, upU_n[: [2, 4[$ $[lowA_{L_n}, upA_{L_n}[: [0, 4[$	$0.63 \leq \mathbb{P}(0)$	$([0, 3[)$	MPS, LPS	0	LPS	0

TAB. 4.2 – *Etats, transitions et sorties d'un décodeur quasi-arithmétique avec* $T = 4$.

bits distribués uniformément, identiquement et indépendamment. Un codeur quasi-arithmétique, en revanche, ne produit pas une séquence de bits distribués uniformément et indépendamment, du fait de l'approximation de la distribution de la source qu'il effectue. Il a été montré dans [HV93] que cette approximation n'entraîne pas une augmentation importante de la longueur du train binaire, si les probabilités des symboles sont comprises entre $1/t$ et $(t-1)/t$, où t est la largeur de l'intervalle courant à subdiviser. Par conséquent, le choix de la précision du codeur quasi-arithmétique T peut dépendre de la distribution de la source. Le débit excédent résultant de l'approximation est calculé de la manière suivante. Soit a un symbole prenant sa valeur dans l'alphabet $\mathcal{A}$, $\mathbb{P}(a)$ la probabilité de l'évènement a et $\mathbb{Q}(a)$ l'approximation de $\mathbb{P}(a)$ faite par le codeur quasi-arithmétique. L'entropie de la source est donnée par

$$H = -\sum_{a=a_1}^{a_M} \mathbb{P}(a) log_2 \mathbb{P}(a). \tag{4.1}$$

L'efficacité du codeur quasi-arithmétique peut alors être mesurée par

$$R = -\sum_{a=a_1}^{a_M} \mathbb{P}(a) log_2 \mathbb{Q}(a). \tag{4.2}$$

Le débit excédent peut alors être exprimé

$$\begin{aligned} E &= R - H, \\ &= \sum_{a=a_1}^{a_M} \mathbb{P}(a) log_2 \frac{\mathbb{P}(a)}{\mathbb{Q}(a)}, \\ &= D(P||Q), \end{aligned} \tag{4.3}$$

où $D(P||Q)$ est la distance de Kullback-Leibler ou entropie relative entre la distribution approchée Q et la distribution exacte P.

4.4 Modèle de source

Pour une source binaire, la variable T peut prendre de petites valeurs (jusqu'à 4), pour une perte modérée en efficacité de compression [HV93]. Cette observation pousse à convertir les sources M-aires en sources binaires avant le codage quasi-arithmétique. Cette conversion est équivalente à un codage binaire à longueur fixe de la source.

4.4.1 Conversion d'une source M-aire en source binaire

Supposons dans un premier temps que la source est une chaîne de Markov d'ordre 0 quantifiée sur $M = 2^q$ symboles. Elle peut être représentée par un arbre binaire de profondeur q, comme cela est montré sur la figure 4.1-a pour $M = 4$. Les probabilités de transition sur les branches de cet arbre binaire sont calculées aisément à partir de la distribution de la source. Cet arbre binaire peut être vu comme un automate qui modélise la distribution stationnaire de la source M-aire. Un modèle complet de la source est obtenu en connectant les *modèles locaux* successifs. Un moyen de le faire consiste à identifier les feuilles de l'arbre binaire avec la racine de l'arbre suivant. Pour une source de Markov d'ordre 0 avec $M = 4$, cela conduit à l'automate à trois états de la figure 4.1-b. Cet automate est mis sous la forme d'un treillis sur la figure 4.1-c, avec les probabilités de chaque bit indiquées sur les transitions correspondantes.

A partir de cet automate stochastique, la séquence de symboles binaires peut être modélisée par une fonction d'un modèle de Markov caché. Soit C_k l'état ν de l'automate après la production de k symboles binaires. La séquence $C_0, \ldots, C_K$ est une chaîne de Markov et la séquence de symboles binaires correspondantes est fonction des transitions de cette chaîne, c'est à dire $S_k = \phi(C_{k-1}, C_k)$.

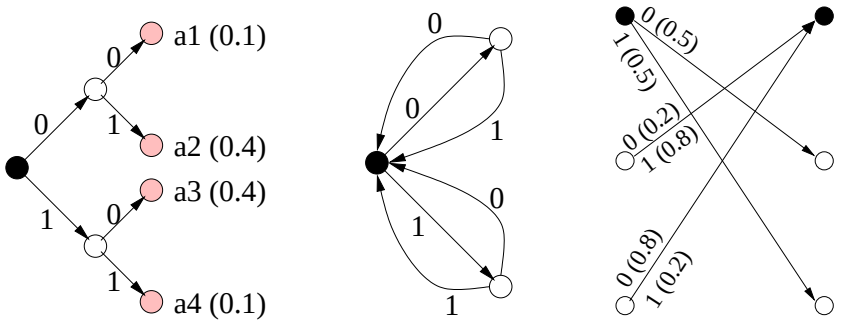


FIG. 4.1 – *Représentation graphique sous la forme d'un a)-arbre, b)-automate stochastique, c)-treillis, du modèle binaire correspondant à une chaîne de Markov d'ordre* 0*, avec* $M = 4$*. Les nœuds noirs correspondent aux feuilles identifiées avec les racines des arbres suivants. Les nœuds blancs correspondent aux nœuds intermédiaires de la représentation binaire des symboles* M*-aires.*

Considérons à présent que la source est une chaîne de Markov d'ordre 1. Pour tenir compte de la corrélation présente dans la source M-aire lors de la construction de la source binaire, il est nécessaire de mémoriser le dernier symbole M-aire codé. Dans ce but, on peut redéfinir la variable d'état C_k par une paire (ν, σ), avec σ la valeur du dernier symbole codé et ν l'état à l'instant bit k de l'automate stochastique décrivant le codage du symbole M-aire suivant. Les probabilités de transition dans cet arbre sont calculées à partir de la loi conditionnelle sur les symboles M-aires, $\mathbb{P}(A_{l+1}|A_l)$. Une autre manière de tenir compte de cette corrélation consiste à connecter $M+1$ arbres de profondeur q, et de définir la variable d'état C_k par un nœud ν dans l'arbre ainsi obtenu.

Un tel arbre est présenté par la figure 4.2-a, pour $M = 4$. Le modèle complet est alors obtenu en identifiant les feuilles de cet arbre avec les nœuds intermédiaires, comme cela est montré sur la figure 4.2-b. Dans ce cas particulier, la taille de l'espace d'états est 15. La taille de l'espace d'états pour une source de Markov d'ordre n quantifiée sur M symboles est $M^{n+1} - 1$. Pour les deux modèles d'états proposés ($C_k = (\nu, \sigma)$ et $C_k = (\nu)$), la séquence $C_1 \dots C_K$ est une chaîne de Markov dont les transitions produisent la séquence de symboles binaires S.

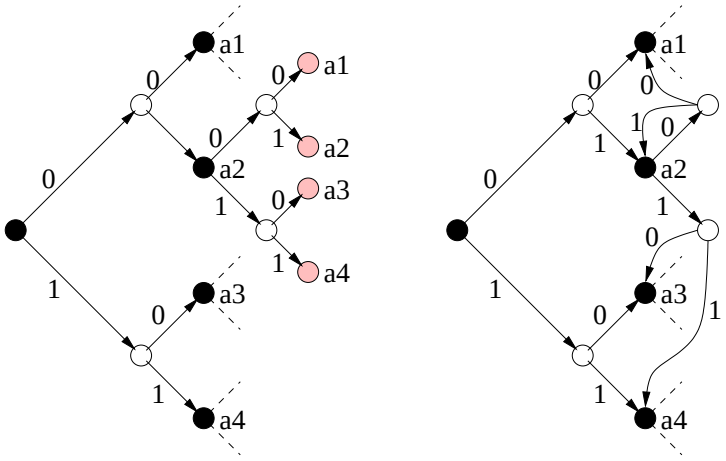


FIG. 4.2 – *Représentation graphique sous la forme d'un a)-arbre, b)-automate stochastique, du modèle binaire correspondant à une chaîne de Markov d'ordre* 1*, avec* $M = 4$*. Les nœuds noirs correspondent aux feuilles identifiées avec les nœuds intermédiaires des arbres suivants. Les nœuds blancs correspondent aux autres nœuds intermédiaires.*

4.4.2 Indexation pour la conversion

Nous avons décrit comment convertir une source M-aire en source binaire à partir des arbres binaires décrivant les mots de codes associés à chacun des symboles M-aires. Le choix des mots de codes binaires associés à chacun des symboles, ou *indexation*, a été ignoré. Nous nous sommes contentés de numéroter les symboles de 0 à $M - 1$, puis de prendre la représentation en base 2 de ces numéros comme mots de codes. Or, cette indexation a une influence sur les performances des procédés d'estimation présentés par la suite. En effet, elle détermine les probabilités de transition à partir de chacun des états du modèle de source binaire. La valeur de ces probabilités reflète l'incertitude sur les transitions quittant chacun des états. Plus l'incertitude est grande sur les transitions quittant un état, plus le procédé d'estimation a de risque de choisir la mauvaise transition. Ce problème de conception d'indexation permettant d'accroître la robustesse d'un code peut apparaître dans d'autres contextes. Il a été étudié par exemple dans le cadre de la quantification vectorielle à bas débit pour la transmission sur des canaux bruités par Zeger et Gersho [ZG90].

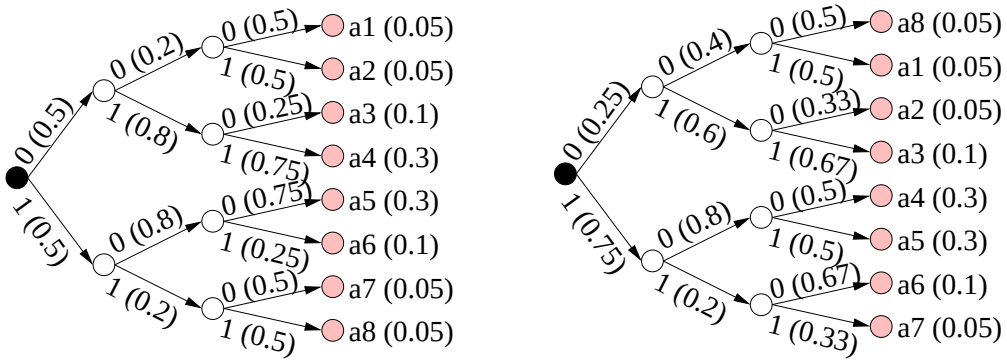


FIG. 4.3 – *Représentation graphique sous la forme d'un arbre du modèle binaire correspondant à une chaîne de Markov d'ordre* 0*, avec* $M = 8$*, a)-sans indexation particulière, b)-avec une indexation basé sur une permutation circulaire d'un symbole. Les probabilités de chaque branche ainsi que les probabilités des feuilles sont indiquées entre parenthèses.*

Soit X un état du modèle de source binaire. L'incertitude sur les transitions quittant cet état est mesurée par l'entropie

$$H(X) = -(\mathbb{P}(0|X)log_2\mathbb{P}(0|X) + \mathbb{P}(1|X)log_2\mathbb{P}(1|X)). \tag{4.4}$$

Plus cette entropie est élevée, plus l'incertitude est grande. Considérons l'exemple de la figure 4.3. Sur la figure 4.3-a, aucune stratégie d'indexation particulière n'a été appliquée. On remarque que l'incertitude associée à la racine de l'arbre est maximale ($H(X) = 1$). Or, la distance entre le symbole décodé et le symbole original est plus importante si ce bit de poids fort est erroné que si c'est un bit de poids faible qui est erroné. L'indexation peut être modifiée simplement en permutant les feuilles de l'arbre décrivant le modèle de source binaire. Sur la figure 4.3-b, une permutation circulaire d'un symbole a été effectuée. On constate une diminution significative de l'incertitude associée à la racine de l'arbre ($H(X) = 0.81$). En revanche, l'incertitude augmente pour d'autres nœuds de l'arbre. Cependant, une erreur bit au niveau de ces nœuds a un impact moindre sur la qualité des symboles décodés. D'une manière générale, nous recherchons et utilisons comme indexation la permutation T^* des feuilles de l'arbre telle que

$$T^* = \arg\min_T \sum_{X_T} \mathbb{P}(X_T)H(X_T). \tag{4.5}$$

Si la taille de l'espace des permutations est trop grande pour une recherche exhaustive, une heuristique efficace consiste à effectuer la recherche uniquement sur les permutations circulaires. Cette recherche de la meilleure indexation n'est pas limitée aux modèles d'ordre 0. Elle s'applique également sur des modèles d'ordre n ou sur des modèles produits tels que ceux présentés dans la section 4.5. Dans ce cas, l'expression

4.4 fait intervenir les probabilités et entropies conditionnelles. On aura tout intérêt à rechercher la permutation optimale pour le modèle utilisé lors de l'estimation.

Nous avons vu dans cette section comment convertir une séquence de symboles M-aires en une séquence de symboles binaires, modélisée par un automate stochastique. Des exemples ont été présentés pour des chaînes de Markov. Cette conversion n'est pas indispensable, mais nous la préconisons pour pouvoir utiliser de petites valeurs de T. Les procédés de décodage présentés par la suite fonctionnent aussi bien pour des sources M-aires que binaires. Cependant, on fera toujours en sorte de modéliser la source par un automate stochastique.

4.5 Automates de codage et de décodage

La conception de procédés efficaces pour estimer la séquence de symboles qui a été transmise nécessite de modéliser les dépendances du train binaire. Dans ce but, nous considérons le produit, dans le sens du produit d'automates des modèles de source binaire et de codeur vus précédemment. Tout ce qui est présenté pour des sources binaires est généralisable aux sources M-aires.

4.5.1 Modèle produit : source + codeur

L'état du modèle produit doit rassembler les informations contenues dans les états de chacun des deux modèles. Par conséquent, l'état X_k du modèle produit est défini par $X_k = (lowS_k, upS_k, C_k)$. On pourrait penser au premier abord que la taille de l'espace d'états résultant de ce produit est donnée par le produit des tailles des espaces d'états des deux modèles (source et codeur). Dans la pratique, des simplifications ont lieu, ce qui permet d'aboutir à un nombre d'états moins important. On peut illustrer ce produit à partir des exemples simples vus précédemment. Le système résultant du produit du modèle de source de Markov d'ordre 0 de la figure 4.1 avec le codeur quasi-arithmétique du tableau 4.1 est donné par la figure 4.4. Pour $T = 4$ et en utilisant la simplification MPS/LPS, il n'y a aucune remise à l'échelle de l'intervalle $[lowS_k, upS_k[$ sans émission de bit. Pour des valeurs supérieures de T, les remises à l'échelle seraient signalées avec la même notation f que dans le tableau 4.1. Dans la mesure ou le modèle de codeur a 2 états et le modèle de source 3, le nombre d'états du modèle produit devrait être 6. Or, il s'avère que l'on obtient seulement 4 états. Cette simplification provient du fait que les transitions du modèle de codeur sont fonction de la distribution de la source. Suivant les probabilités stationnaires de la source binaire, le modèle général de codeur présenté dans le tableau 4.1 se simplifie en deux codeurs montrés sous forme de treillis par la figure 4.5. Les probabilités de la source binaire résultant d'une conversion à partir d'une source M-aire dépendent de l'état C_k du modèle de cette source binaire. Par conséquent, les deux modèles de codeur peuvent être utilisés successivement, suivant l'état du modèle de la source binaire. Dans l'exemple de la figure 4.5-b, on vérifie aisément que l'état $X_k = 1$ ne peut pas être atteint, d'où la réduction du nombre d'états du modèle produit.

State	State Variables	MPS	LPS
0 (start)	[0,4[C=0	output : 0 next state : 1	output : 1 next state : 2
1	[0,4[C=1	output : − next state : 3	output : 11 next state : 0
2	[0,4[C=2	output : − next state : 3	output : 11 next state : 0
3	[0,3[C=0	output : 0 next state : 1	output : 10 next state : 2

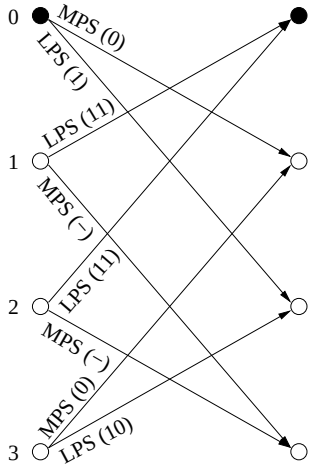


FIG. 4.4 – *Modèle produit source + codeur : a) états, sorties et transitions; b) Représentation en treillis.*

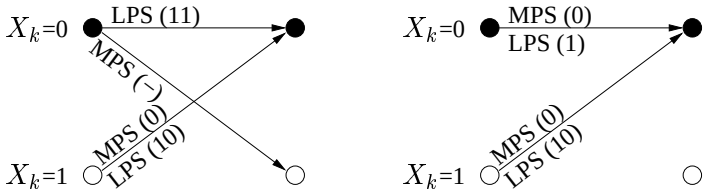


FIG. 4.5 – *Modèles de codeur quasi-arithmétique ($T = 4$) : a) $0.63 \leq \mathbb{P}(MPS)$; b) $0.5 \leq \mathbb{P}(MPS) < 0.63$.*

Dans le cas général, la taille de l'espace d'états ne peut pas être connue a priori. Elle doit être calculée à partir du modèle de source binaire et du paramètre T. Soit N_C le nombre d'états du modèle de source binaire, le nombre maximum d'états du modèle produit est $\frac{3T^2 N_C}{16}$.

Comme le nombre de bits émis par chaque transition du modèle produit est aléatoire, la structure des dépendances entre la séquence de mesures et les états du modèle est aléatoire. Afin de "capturer" cette structure de dépendances, nous considérons le processus de Markov augmenté $(X, N) = (X_1, N_1) \dots (X_K, N_K)$. Cela conduit à la structure de dépendances décrite graphiquement par la figure 4.6. Suivant ce modèle, une séquence de symboles binaires S_1^K est convertie en une séquence de bits $U_1^{N_K}$, avec N_K le nombre de bits émis quand K symboles ont été codés. Etant donnés un état X_k et un symbole d'entrée S_{k+1}, l'automate spécifie les bits $\bar{U}_{k+1} = U_{N_k+1}^{N_{k+1}}$ qui doivent être émis et l'état suivant X_{k+1}. On remarque que certaines transitions n'émettent aucun bit. Les probabilités des transitions entre $(X_k, N_k) = (lowS_k, upS_k, C_k, N_k)$ et $(X_{k+1}, N_{k+1}) = (lowS_{k+1}, upS_{k+1}, C_{k+1}, N_{k+1}))$ dans le treillis sont données par le modèle de source binaire, $\mathbb{P}(C_{k+1}|C_k) = \mathbb{P}(S_{k+1}|C_k))$. Les mesures $\bar{Y}_k$ sur les bits $\bar{U}_k$ sont obtenues à la sortie du canal de transmission.

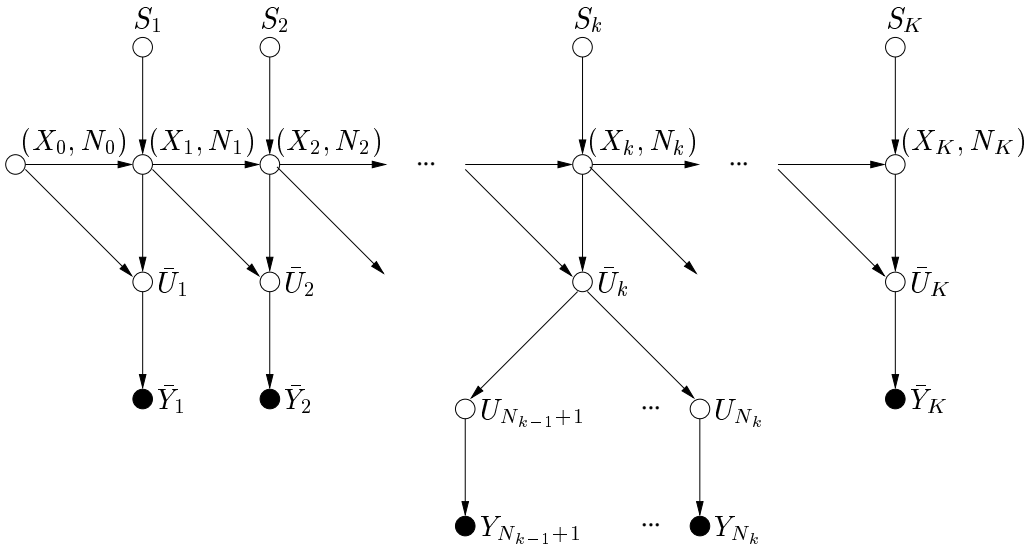


FIG. 4.6 – *Structure de dépendances du modèle produit source + codeur ($N_k \geq N_{k-1}$). Les états blancs et noirs représentent respectivement les variables cachées et observées.*

4.5.2 Modèle produit : source + décodeur

Un modèle produit de l'ensemble source + décodeur peut être construit de la même manière. Les états de ce modèle produit doivent rassembler les informations des états des modèles de source et de décodeur. Un état X_n de ce modèle produit est donc défini par $X_n = (lowU_n, upU_n, lowS_{K_n}, upS_{K_n}, C_{K_n})$. Le système résultant du produit du

décodeur du tableau 4.2 avec le modèle de source simple de la figure 4.1-c est donné par la figure 4.7. Là encore, la taille de l'espace d'états dépend de la précision T du codeur et du modèle de source binaire.

State	State Variables	0	1
0 (start)	[0,4[[0,4[C=0	output : MPS next state : 1	output : LPS next state : 2
1	[0,4[[0,4[C=1	output : MPS,MPS next state : 1	output : − next state : 3
2	[0,4[[0,4[C=2	output : MPS,MPS next state : 1	output : − next state : 4
3	[2,4[[0,4[C=1	output : MPS,LPS next state : 2	output : LPS next state : 0
4	[2,4[[0,4[C=2	output : MPS,LPS next state : 2	output : LPS next state : 0

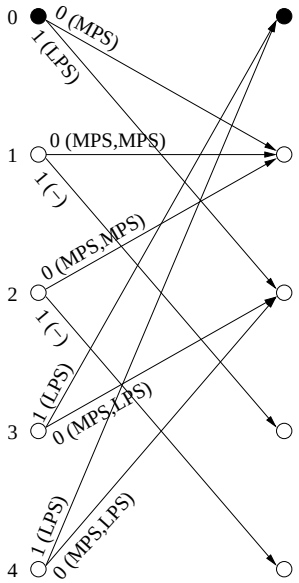


FIG. 4.7 – *Modèle produit source + décodeur : a) états, sorties et transitions; b) Représentation en treillis.*

Le nombre de symboles produits par chaque transition du modèle produit est aléatoire. Par conséquent, la structure des dépendances entre la séquence de mesures et les symboles décodés est aléatoire. Nous traitons cette difficulté en considérant la chaîne de Markov augmentée $(X, K) = (X_1, K_1) \dots (X_N, K_N)$, dont la structure des dépendances est illustrée graphiquement par la figure 4.8. Suivant ce modèle, une séquence de bits U_1^N est convertie en une séquence de symboles $S_1^{K_N}$, avec K_N le nombre de symboles décodé quand N bits ont été reçus. Etant donnés un état X_n et un bit d'entrée U_{n+1}, l'automate produit la séquence de symboles $\bar{S}_{n+1} = S_{K_n+1}^{K_{n+1}}$, et spécifie l'état suivant X_{n+1}. Les probabilités des transitions entre (X_n, K_n) et (X_{n+1}, K_{n+1}) dans le treillis dépendent du modèle de source binaire. Elles sont données par $\prod_{k=K_n+1}^{K_{n+1}} \mathbb{P}(S_k|C_{k-1})$.

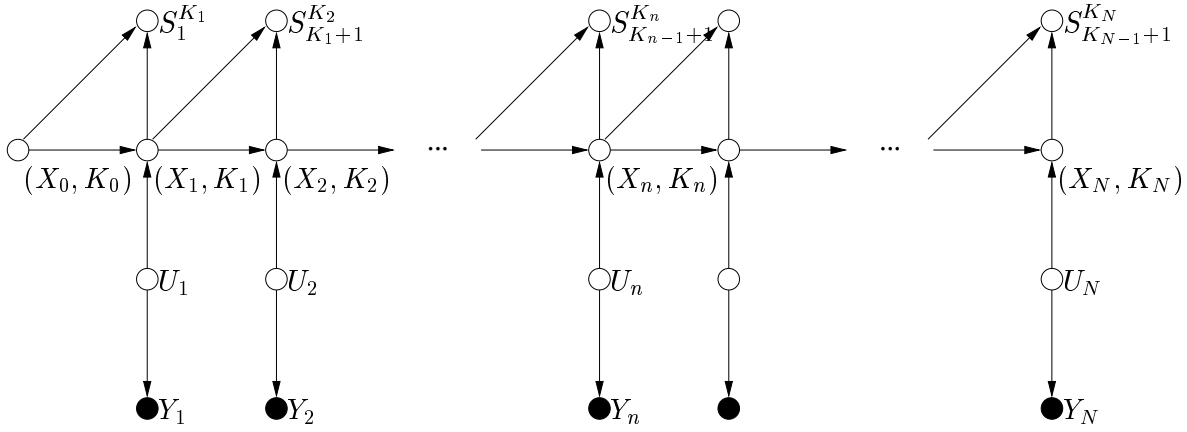


FIG. 4.8 – *Structure de dépendances du modèle produit source+décodeur. Les états blancs et noirs représentent respectivement les variables cachées et observées.*

4.5.3 Approximation de la distribution de la source

L'entropie d'une source M-aire peut être calculée à partir du modèle de source binaire correspondant (section 4.4.1). Elle est donnée par

$$H(S|C) = -\sum_{c=0}^{N_C-1} \mathbb{P}(c) \sum_{s=0,1} \mathbb{P}(s|c) log_2 \mathbb{P}(s|c), \tag{4.6}$$

où C représente la variable d'état du modèle de source binaire, c les indices des valeurs possibles de cette variable et N_C le nombre d'états possibles. L'efficacité d'un codeur quasi-arithmétique appliqué sur ce modèle de source pour une valeur donnée du paramètre T peut être mesurée par

$$R(S|X) = -\sum_{x=0}^{N_X-1} \mathbb{P}(x) \sum_{s=0,1} \mathbb{P}(s|x) log_2 \mathbb{Q}(s|x), \tag{4.7}$$

où X représente la variable d'état du modèle produit (source + (dé)codeur), x les indices des valeurs possibles de cette variable et N_X le nombre d'états possibles de ce modèle produit. Le codeur quasi-arithmétique effectue une approximation de la distribution de la source binaire et donc de la source M-aire. Cependant, pour une valeur de T fixée, l'approximation est plus précise quand on applique le codeur sur la source binaire que quand on l'applique directement sur la source M-aire. Dans le cas du modèle produit, le débit excédent pour une valeur de T donnée est

$$\begin{aligned} E(S|X) &= R(S|X) - H(S|X), \\ &= \sum_{x=0}^{N_X-1} \mathbb{P}(x) \sum_{s=0,1} \mathbb{P}(s|x) log_2 \frac{\mathbb{P}(s|x)}{\mathbb{Q}(s|x)}, \\ &= D(\mathbb{P}(s|x)||\mathbb{Q}(s|x)), \end{aligned} \tag{4.8}$$

où $D(\mathbb{P}(s|x)||\mathbb{Q}(s|x))$ est l'entropie relative conditionnelle entre la distribution conditionnelle approchée $\mathbb{Q}$ et la distribution conditionnelle exacte $\mathbb{P}$. On remarque que plusieurs états du codeur quasi-arithmétique peuvent correspondre à un état donné c du modèle de source binaire. Par conséquent, plusieurs approximations $\mathbb{Q}(s|x)$ de $\mathbb{P}(s|c)$ peuvent exister (voir par exemple les états 0 et 3 sur la figure 4.4). Le débit excédent peut être considéré comme une mesure de la redondance du train binaire.

4.5.4 Codage quasi-arithmétique adaptatif

Il a été mentionné en section 3.7 que les algorithmes d'estimation proposés en section 3.5 restent valide dans le cas du codage arithmétique adaptatif.

Dans le cas du codage quasi-arithmétique, l'adaptabilité est bien évidemment possible. Cependant, son utilisation est en contradiction avec la modélisation que nous avons adoptée. En effet, dans cette modélisation, l'aspect adaptatif intervient au niveau du modèle de source. Ce dernier est représenté sous la forme d'un automate stochastique dont tous les états sont *connus a priori*. C'est cette notion de connaissance a priori qui est contradictoire avec l'adaptabilité. Si la structure contenant les statistiques de la source mises à jour dynamiquement a un nombre d'états fini, alors il est envisageable de calculer un modèle de source a priori qui inclurait toutes les statistiques de la source possibles. Cependant, on comprend aisément que le nombre d'états d'un tel modèle risque d'être prohibitif. Il semble donc préférable de limiter l'utilisation du décodage souple de codes quasi-arithmétiques à des modèles de sources stationnaires. On peut cependant imaginer des stratégies intermédiaires ou l'adaptabilité est limitée. En particulier, il est possible de définir un ensemble fini de modèles de sources en s'appuyant sur les approximations qui sont faites par le codeur quasi-arithmétique. Par exemple, pour une source binaire et une précision de $T = 4$, seuls 3 modèles de source approchés existent. Cependant, cette approche pourrait nuire à la fiabilité de l'algorithme d'estimation présenté ci-dessous. En effet, celui-ci exploite la distribution réelle de la source, et non la distribution approchée utilisée par le codeur.

Nous préférons conserver la méthode d'estimation du chapitre précédent pour traiter le cas du codage adaptatif. Nous proposons à présent un algorithme d'estimation pour le décodage de codes quasi-arithmétiques.

4.6 Estimation

Dans cette section, nous proposons un procédé d'estimation pour le décodage de codes quasi-arithmétiques basé sur les modèles de dépendances décrits précédemment. Le critère du MAP (*maximum a posteriori*) correspond à l'estimation bayésienne optimale d'un processus X à partir des observations disponibles Y :

$$\hat{X} = \arg\max_{x} \mathbb{P}(X = x|Y). \tag{4.9}$$

L'optimisation est effectuée pour toutes les *séquences* x possibles. Nous appliquons ce critère à l'estimation des états cachés des processus (X, N) (horloge symbole) et (X, K)

(horloge bit) étant données les séquences de bits observées. Pour éviter le problème de la gestion de la variable n_{scl}, nous considérons uniquement l'utilisation du modèle produit source + décodeur.

L'estimation de l'ensemble des états cachés $(X, K) = (X_1, K_1) \dots (X_N, K_N)$ est équivalente à l'estimation de la séquence de symboles décodés correspondante $S = S_1 \dots S_{K_n} \dots S_{K_N}$, sachant les observations Y_1^N à la sortie du canal. La meilleure séquence (X, K) peut être obtenue à partir des probabilités locales sur les paires (X_n, K_n) grâce à l'équation

$$\mathbb{P}(X, K|Y) = \prod_{n=1}^{N} \mathbb{P}(X_n, K_n|Y). \tag{4.10}$$

Le calcul de $\mathbb{P}(X_n, K_n|Y)$ peut être effectué à partir de la décomposition

$$\mathbb{P}(X_n, K_n|Y) \propto \mathbb{P}(X_n, K_n|Y_1^n) \cdot \mathbb{P}(Y_{n+1}^N|X_n, K_n), \tag{4.11}$$

où $\propto$ désigne un facteur de normalisation. Les dépendances de la chaîne de Markov permettent un calcul récursif des deux termes de la partie droite de cette équation, basé sur l'algorithme BCJR [BCJR74]. Cet algorithme est présenté en annexe C. La passe avant permet de calculer le premier terme

$$\begin{aligned}\mathbb{P}(X_n = x_n, K_n = k_n|Y_1^n) = \sum_{(x_{n-1},k_{n-1})} & \mathbb{P}(X_{n-1} = x_{n-1}, K_{n-1} = k_{n-1}|Y_1^{n-1}) \cdot \\ & \mathbb{P}(X_n = x_n, K_n = k_n|X_{n-1} = x_{n-1}, K_{n-1} = k_{n-1}) \cdot \\ & \mathbb{P}(U_n = u_{(x_{n-1},k_{n-1})(x_n,k_n)}|Y_n).\end{aligned} \tag{4.12}$$

Les termes de cette équation sont respectivement le terme récursif, la probabilité de transition donnée par le modèle produit (cf section 4.5) et la probabilité d'avoir émis le bit U_n déclenchant la transition entre $X_{n-1} = x_{n-1}$ et $X_n = x_n$, sachant la mesure Y_n. Ce dernier terme est obtenu à partir du modèle de canal. Le calcul récursif est initialisé à l'état de départ $(0, 0)$ et permet de calculer $\mathbb{P}(X_n, K_n|Y_1^n)$ pour tout état possible (x_n, k_n) à chaque instant $n = 1, \dots N$ de l'horloge bit.

La passe arrière permet d'obtenir le second terme de l'équation 4.11

$$\begin{aligned}\mathbb{P}(Y_{n+1}^N|X_n = x_n, K_n = k_n) = \sum_{(x_{n+1},k_{n+1})} & \mathbb{P}(Y_{n+2}^N|X_{n+1} = x_{n+1}, K_{n+1} = k_{n+1}) \cdot \\ & \mathbb{P}(X_{n+1} = x_{n+1}, K_{n+1} = k_{n+1}|X_n = x_n, K_n = k_n) \cdot \\ & \mathbb{P}(U_{n+1} = u_{(x_n,k_n)(x_{n+1},k_{n+1})}|Y_{n+1}).\end{aligned} \tag{4.13}$$

Le calcul récursif est initialisé pour tous les "derniers états" possibles (x_N, k_N) et permet de calculer $\mathbb{P}(Y_{n+1}^N|X_n, K_n)$ pour tout état possible (x_n, k_n) à chaque instant bit n consécutivement de N à 1.

Le treillis sur lequel l'estimation est effectuée peut être pré-calculé et mémorisé à partir du modèle produit source + décodeur et de la longueur N de la séquence de

bits observée. Lorsque le nombre de symboles est connu, cela permet de rajouter une contrainte de terminaison : tous les chemins dans le treillis qui ne conduisent pas au bon nombre de symboles ($K_N = K$) sont supprimés. L'utilisation de cette contrainte de terminaison nécessite de construire le treillis en deux passes. La première démarre à l'état initial du treillis et construit successivement les différentes section du treillis. Au cours de cette passe avant, il n'est pas possible de savoir quels sont les chemins qui ne respectent pas la contrainte de terminaison. Une fois la passe avant terminée, on peut supprimer les états de la dernière section ($n = N$) du treillis qui ne respectent pas la contrainte. Une passe arrière permet alors de supprimer toutes les branches qui n'aboutissent pas. La figure 4.9 montre le treillis d'estimation calculé à partir du modèle produit de la figure 4.7 pour une séquence de $K = 7$ symboles produisant $N = 6$ bits.

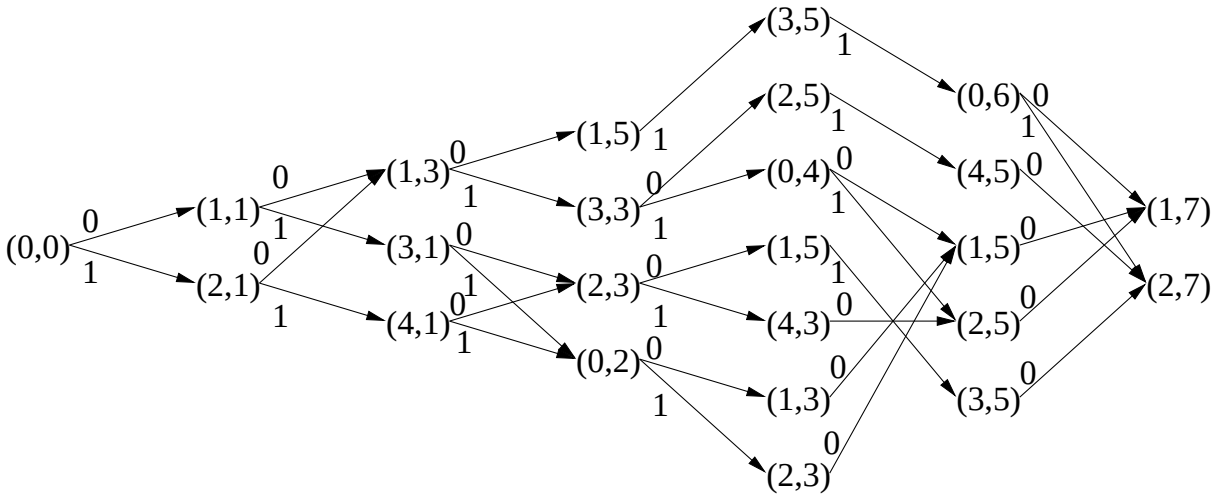


FIG. 4.9 – *Exemple de treillis pour* $K = 7$ *et* $N = 6$.

Le nombre d'états du treillis dépend de *l'excursion* de la variable K_n, c'est à dire de la plage de valeurs qu'elle peut prendre pour un index n donné. Si K et N sont connus, ce treillis est alors plus "large" au milieu qu'à ses extrémités. Le nombre d'états du treillis ne peut pas être calculé a priori, mais peut être majoré. Lorsque le nombre d'états du treillis devient trop important, une solution possible consiste à effectuer l'estimation avec l'algorithme de la section 3.5. Ce dernier s'applique naturellement aux codes quasi-arithmétiques. La complexité est alors contrôlée par élagage. Les marqueurs de synchronisation présentés en section 4.7.1 permettent également de réduire la taille du treillis.

Finalement, la structure de l'algorithme d'estimation est résumée par la figure 4.10. Une variante de cette structure, où le treillis sur lequel est faite l'estimation est calculé séparemment, est présentée sur la figure 4.11.

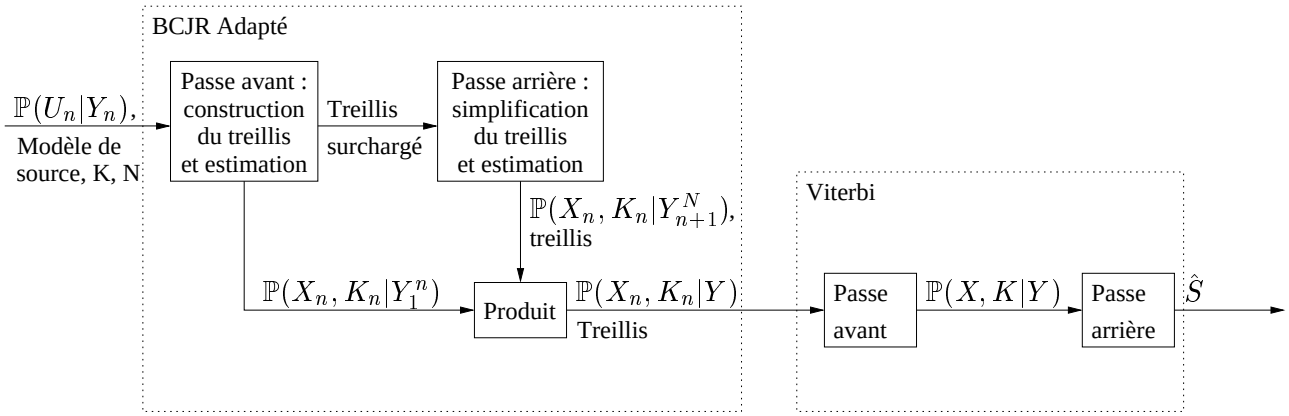


FIG. 4.10 – *Procédé d'estimation en deux passes sur treillis.*

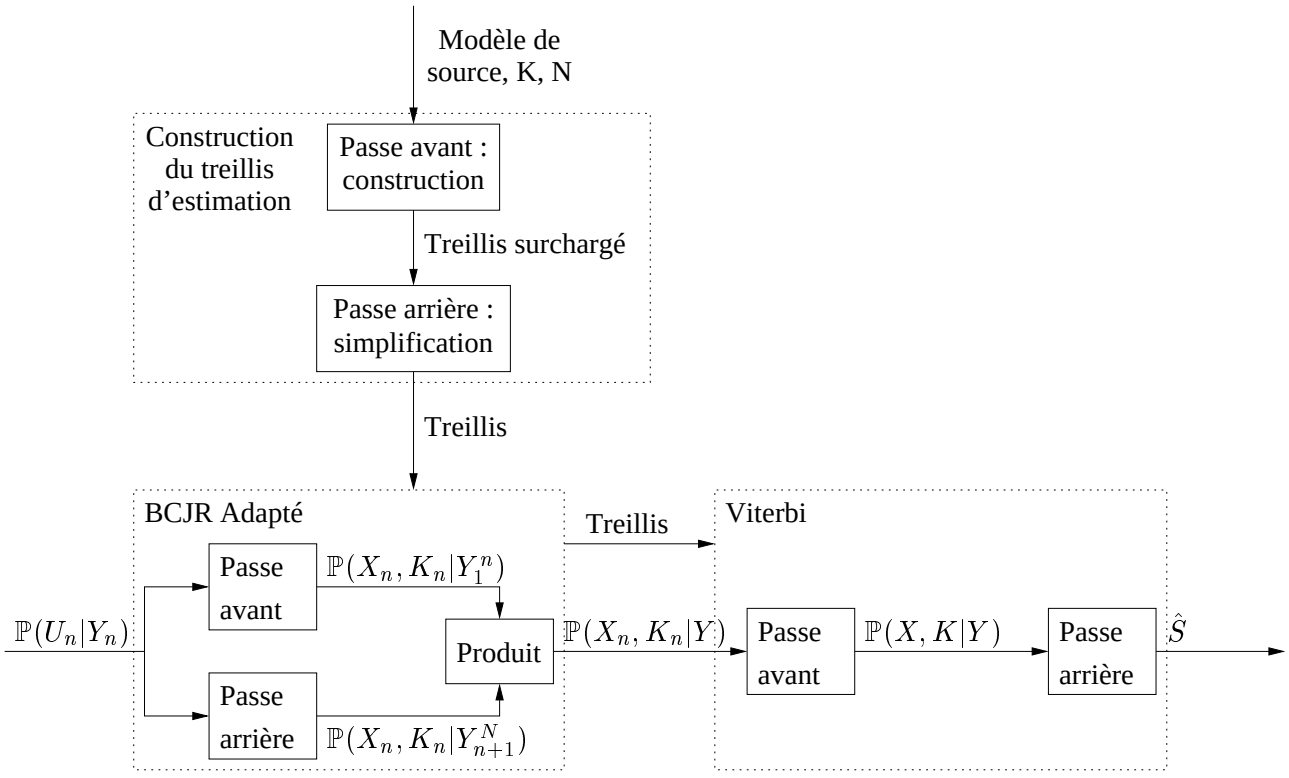


FIG. 4.11 – *Procédé d'estimation en deux passes sur treillis, construction du treillis séparée de l'estimation.*

4.7 Amélioration des performances par ajout de redondance

4.7.1 Marqueurs de synchronisation

Les contraintes de terminaison mentionnées en section 4.6 peuvent être vues comme un moyen de forcer la synchronisation à la fin de la séquence. En effet, elles contraignent le décodeur à obtenir la bonne correspondance entre le nombre de symboles décodés et le nombre de bits observés. Cependant, ces contraintes n'ont pas d'effet de synchronisation sur le milieu de la séquence. Nous proposons l'addition à la séquence de symboles d'information complémentaire afin de favoriser la re-synchronisation sur l'ensemble de la séquence.

Pour cela, nous introduisons des symboles supplémentaires, ou marqueurs, à des positions connues $I_s = \{i_1, \ldots, i_s\}$ dans la séquence de symboles. Le choix de ces symboles est arbitraire, sur le principe des techniques décrites dans [BCI+97], [DHS01], [PHS01], [Elm99a] et [SCW00]). Nous proposons soit de choisir un des symboles dans l'alphabet, soit de répéter le dernier symbole codé. Ces marqueurs sont insérés à position connue suivant l'horloge symbole. Par conséquent, la position des bits supplémentaires correspondants dans le train binaire dépend de la séquence de symboles codée, donc est aléatoire.

Les procédés développés précédemment doivent tenir compte de cette information supplémentaire. Insérer un marqueur dans la séquence de symboles revient à ajouter une ou plusieurs *sections* avec des transitions déterministes dans le modèle de source binaire. Cette connaissance a priori peut être exploitée par les procédés d'estimation. La variable K_n (k à horloge symbole) indique quand un marqueur est attendu. Les probabilités de transition correspondantes dans le treillis d'estimation sont mises à jour en conséquence. Une probabilité nulle est affectée à toutes les transitions qui n'émettent pas les marqueurs attendus, alors qu'une probabilité de un est affectée aux autres. Par conséquent, certains chemins dans le treillis peuvent être supprimés, ce qui amène une réduction du nombre d'états et une meilleure capacité de re-synchronisation.

4.7.2 Information adjacente

De la même manière qu'en section 3.8.2, nous pouvons améliorer la capacité de resynchronisation de l'algorithme d'estimation en transmettant de l'information adjacente en parallèle du train binaire. Ainsi la transmission de la valeur du compteur K_n à intervalles réguliers peut être exploitée lors de la construction du treillis d'estimation, ce qui conduit en plus à une réduction de la taille de ce dernier. Cependant, tout comme dans la section 3.8.2, la nécessité de protéger fortement cette information adjacente entraîne un coût de codage qui limite son utilisation.

4.7.3 Décodage turbo itératif

Les mécanismes de resynchronisation décrits ci-dessus permettent d'augmenter significativement la fiabilité de l'estimation. On peut également considérer l'utilisation

d'un code correcteur d'erreur, de la même manière que dans la section 3.8.3. La différence avec le schéma décrit en section 3.8.3 réside dans le fait que l'estimation de codes quasi-arithmétiques ne nécéssite pas d'élagage. On s'attend donc à de meilleurs résultats. Nous rappelons ici le fonctionnement de ce schéma adapté à l'utilisation du codage quasi-arithmétique. Pour une justification théorique de l'utilisation d'un tel schéma, se reporter à [Guy02].

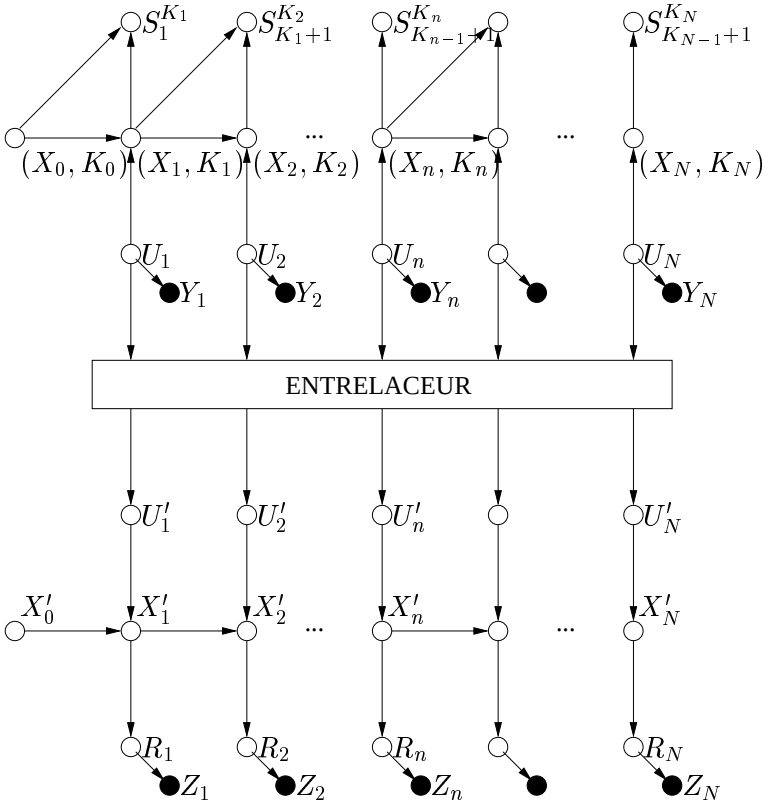


FIG. 4.12 – *Représentation graphique à horloge bit des dépendances dans la chaîne de codage conjoint arithmétique-codage de canal. Les points blancs et noirs sont respectivement les variables cachées et observées.*

Nous proposons d'utiliser un code convolutif systématique (CC). Ce code peut être concaténé avec le code quasi-arithmétique, dans l'esprit des turbo codes en série. A partir de ce principe, on peut utiliser les deux modèles séparément dans un schéma itératif, à condition de les séparer par un entrelaceur [GFGR01]. La structure des dépendances entre les variables d'un tel schéma est représentée graphiquement sur la figure 4.12. A partir de la séquence de bits U_1^N, le codeur de canal dont les états sont notés X' génère la séquence de bits de redondance R_1^N. Les observations Z_1^N de ces bits sont disponibles à la sortie du canal. Le rendement du code de canal est controlé par poinçonnage.

Un tel schéma nécessite de transmettre l'information extrinsèque sur les bits U_n du décodeur de codes convolutifs vers le décodeur arithmétique souple, et réciproquement.

Cette information représente la modification induite sur la loi conditionnelle de U_n sachant Y_n par l'introduction du reste des observations Y_1^{n-1}, Y_{n+1}^N. Cette information extrinsèque peut s'exprimer

$$Ext_{U_n}(Y|Y_n) \propto \frac{\mathbb{P}(U_n|Y)}{\mathbb{P}(U_n|Y_n)}. \tag{4.14}$$

L'estimation itérative consiste tout d'abord à appliquer un algorithme BCJR [BCJR74] sur le code de canal. L'information extrinsèque sur les bits U_n est directement produite par cet algorithme. Elle sert d'entrée pour l'estimation effectuée sur le modèle de décodeur quasi-arithmétique. Le résultat de cette estimation est constitué des probabilités a posteriori sur les états (X_n, K_n) du treillis d'estimation sachant les observations : $\mathbb{P}(X_n, K_n|Y)$. Ces probabilités a posteriori doivent être converties en information extrinsèque sur les bits U_n. La probabilité d'avoir un bit $U_n = u_n$ sachant Y est la somme des probabilités de toutes les transitions entre les états (x_{n-1}, k_{n-1}) et (x_n, k_n) pour lesquels cette transition existe et est déclenchée par u_n. Ainsi la probabilité a posteriori sur le bit U_n sachant les observations est donnée par

$$P(U_n = u_n|Y)|_{u_n=0,1} = \sum_{(x_{n-1},k_{n-1})} \mathbb{P}(x_{n-1}, k_{n-1}|Y) \cdot \frac{\mathbb{P}(succ_{u_n}(x_{n-1}, k_{n-1})|Y)}{\sum_{u_n=0,1} \mathbb{P}(succ_{u_n}(x_{n-1}, k_{n-1}|Y))}, \tag{4.15}$$

où $succ_{u_n}(x_{n-1}, k_{n-1})$ est l'état (x_n, k_n) atteint par la transition quittant (x_{n-1}, k_{n-1}) qui est déclenchée par le bit $U_n = u_n$.

4.7.4 Résultats expérimentaux

Afin d'évaluer les performances du décodage souple de codes quasi-arithmétiques, une série d'expériences a été réalisée sur une source de Gauss-Markov d'ordre 1 de moyenne nulle et de variance unité avec différents facteurs de corrélation ρ. La source est quantifiée uniformément sur 8 niveaux (3 bits) dans l'intervalle $[-3, 3]$ et nous utilisons des séquences de $K = 200$ symboles. Toutes les simulations ont été effectuée pour un canal gaussien à bruit blanc additif (AWGN) avec une modulation BPSK. Les résultats sont moyennés sur 400 réalisations.

La première série d'expériences permet de comparer les performances en termes de taux d'erreur symbole (SER) et de rapport signal à bruit (SNR) de l'algorithme de décodage souple de codes quasi-arithmétiques, pour $T = 4$, avec le décodage souple de codes arithmétiques et le décodage souple de codes de Huffman [GFGR01]. La comparaison est effectuée pour des débits comparables, ceux-ci étant ajustés avec des marqueurs de synchronisation quand cela est nécessaire.

Les figures 4.13 et 4.14 présentent les courbes de SER et SNR résiduels en fonction du rapport signal à bruit E_b/N_0 du canal, respectivement pour $\rho = 0.5$ et $\rho = 0.9$. Pour des sources faiblement corrélées ($\rho = 0.5$ et en dessous), les codes quasi-arithmétiques surpassent les codes arithmétiques et les codes de Huffman. Pour les sources fortement corrélées, en revanche, les codes quasi-arithmétiques sont moins efficaces que les

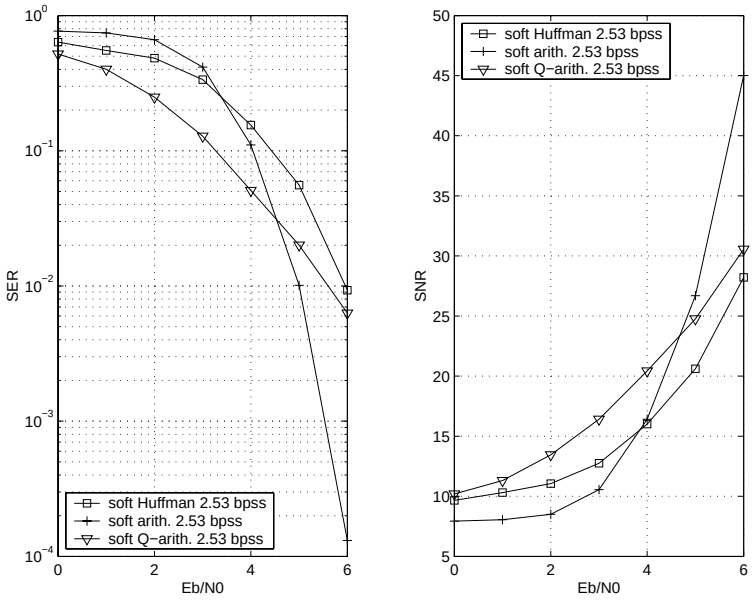


FIG. 4.13 – *SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques* ($\rho = 0.5$, 200 *symboles*, 400 *réalisations*).

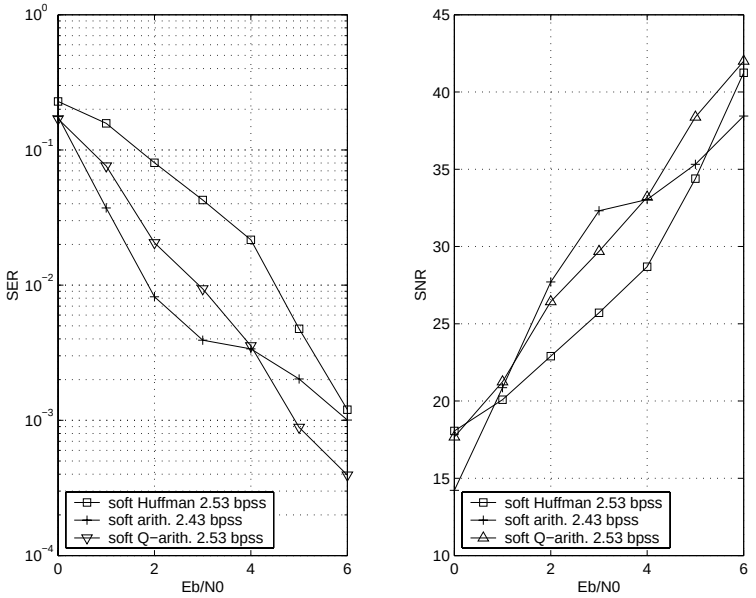


FIG. 4.14 – *SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques* ($\rho = 0.9$, 200 *symboles*, 400 *réalisations*).

codes arithmétiques, mais restent supérieurs aux codes de Huffman. Cela s'explique par le fait que pour les fortes corrélations, les probabilités utilisées par le codeur quasi-arithmétique peuvent avoir des valeurs très faibles ou très élevées. Or, on a vu que ce cas est défavorable et entraîne un débit excédent plus important (section 4.3.3). Ce phénomène est d'autant plus marqué que nous utilisons la précision minimale $T = 4$. Par conséquent, la perte en compression limite l'ajout de redondance, ce qui défavorise les codes quasi-arithmétiques par rapport aux codes arithmétiques. Cette différence se réduit lorsqu'on augmente la précision T. Les codes quasi-arithmétiques conservent quoi qu'il en soit une efficacité supérieure aux codes de Huffman. La même tendance est observée pour un ajout de redondance plus important. Les figures 4.15 et 4.16, présentent les résultats obtenus en ajoutant des marqueurs de synchronisation aux trois méthodes de codage et estimation.

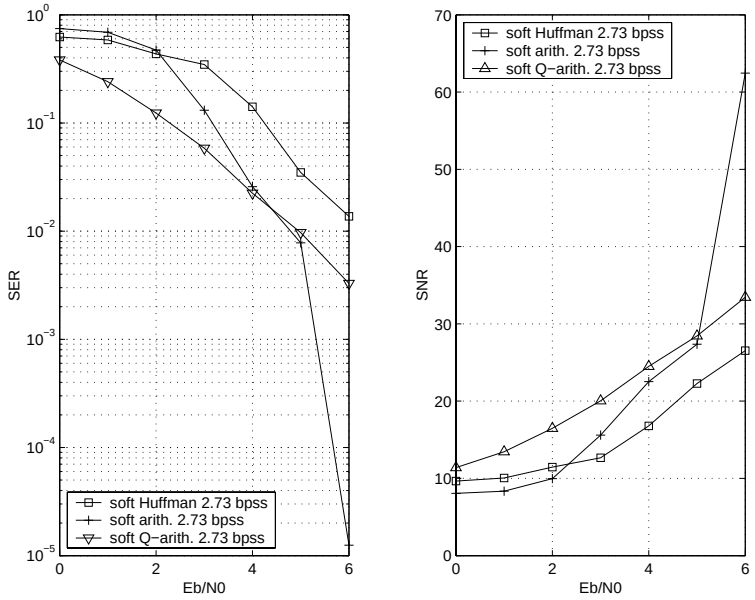


FIG. 4.15 – *SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques, avec synchronisation souple ($\rho = 0.5$, 200 symboles, 400 réalisations).*

Comme le montre la figure 4.17, il est préférable d'utiliser des marqueurs de synchronisation plutôt que de l'information adjacente pour augmenter la robustesse de l'estimation. Du fait de son coût plus important, l'information adjacente est ajoutée en moins grande quantité que les marqueurs. Les performances des deux techniques en terme de temps de calcul et de taille des treillis d'estimation sont équivalentes.

La seconde série d'expériences permet d'évaluer les performances du schéma de décodage itératif, en le comparant à l'ajout de marqueurs de synchronisation. Les figures 4.18 et 4.19 présentent les courbes de SER et SNR résiduels en fonction du rapport signal à bruit E_b/N_0 du canal, respectivement pour $\rho = 0.5$ et $\rho = 0.9$. La première

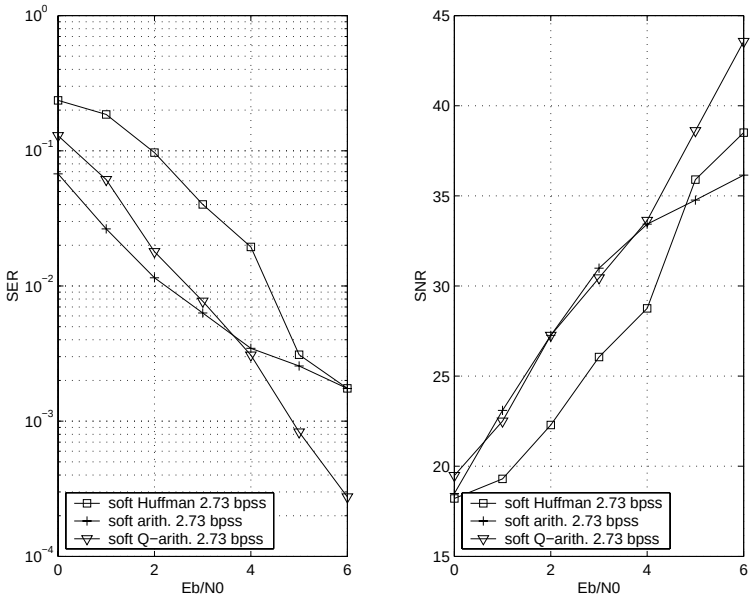


FIG. 4.16 – *SER et SNR pour (a) le décodage souple de codes de Huffman (b) le décodage souple de codes arithmétiques et (c) le décodage souple de codes quasi-arithmétiques, avec synchronisation souple* ($\rho = 0.9$, 200 *symboles*, 400 *réalisations).*

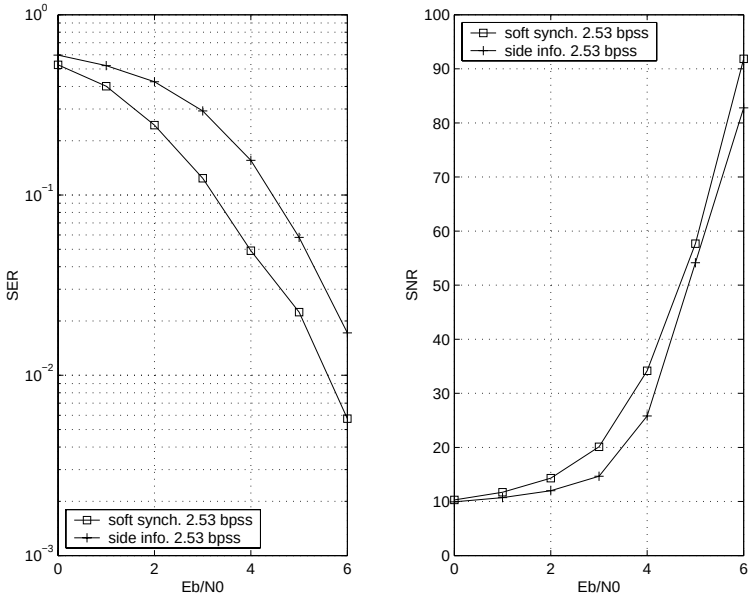


FIG. 4.17 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques (a) avec synchronisation souple et (b) avec information adjacente* ($\rho = 0.5$, 200 *symboles*, 400 *réalisations).*

observation est que les itérations apportent un gain significatif. Nous obtenons donc bien le résultat attendu (section 4.7.3). L'efficacité du schéma itératif dépend de la corrélation de la source. Pour des sources peu corrélées, la synchronisation souple se révèle être plus efficace en terme de SER, et en terme de SNR pour $E_b/N_0 \leq 2$ dB. Dans la mesure où le code convolutif ne peut pas corriger la totalité des erreurs, le problème de désynchronisation du code à longueur variable reste prédominant. Il est plus efficace de le traiter spécifiquement. Quand la corrélation de la source est élevée, au contraire, le schéma itératif se montre nettement plus efficace que les marqueurs de synchronisation. La présence d'une forte corrélation entre les symboles favorise l'algorithme d'estimation. Les marqueurs de synchronisation deviennent donc moins importants. Le code convolutif permet alors de réduire le taux d'erreur binaires à l'entrée de l'estimation, ce qui favorise encore celle-ci. On a une meilleure complémentarité entre les deux éléments du schéma itératif.

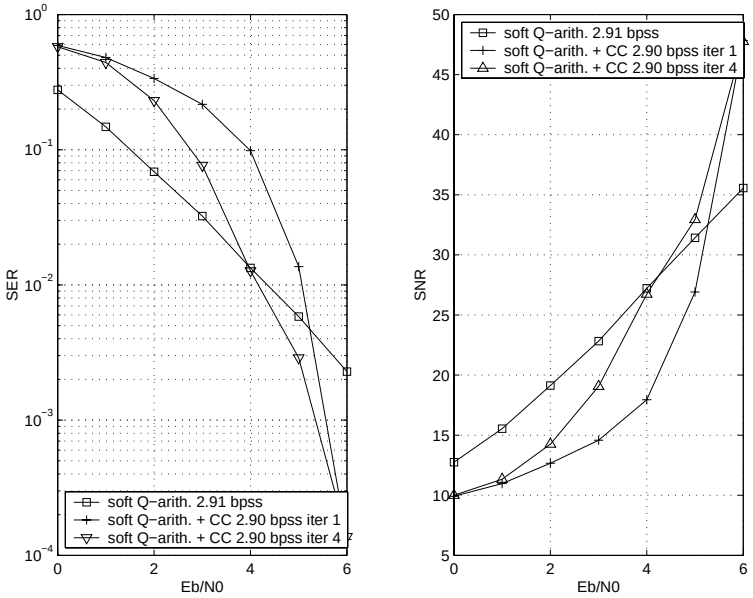


FIG. 4.18 – *SER et SNR pour (a) le décodage souple de codes quasi-arithmétiques et (b) le décodage turbo itératif de codes quasi-arithmétiques* ($\rho = 0.5$, 200 *symboles,* 400 *réalisations).*

L'expérience suivante permet d'évaluer l'impact du réglage de la précision du codeur quasi-arithmétique sur l'efficacité de l'estimation. La figure 4.20 présente les courbes de SER et SNR résiduels en fonction de E_b/N_0 ($\rho = 0.5$), pour $T = 4$ sans redondance ajoutée, $T = 8$ sans redondance ajoutée et $T = 8$ avec redondance ajoutée jusqu'à un débit comparable au cas où $T = 4$. Le codeur de précision inférieure ($T = 4$) est plus robuste aux erreurs, du fait de la présence de redondance résiduelle plus importante. Cependant, le codeur plus précis atteint une meilleure efficacité en compression, grâce à sa meilleure exploitation des statistiques de la source. Lorsque ce gain en compres-

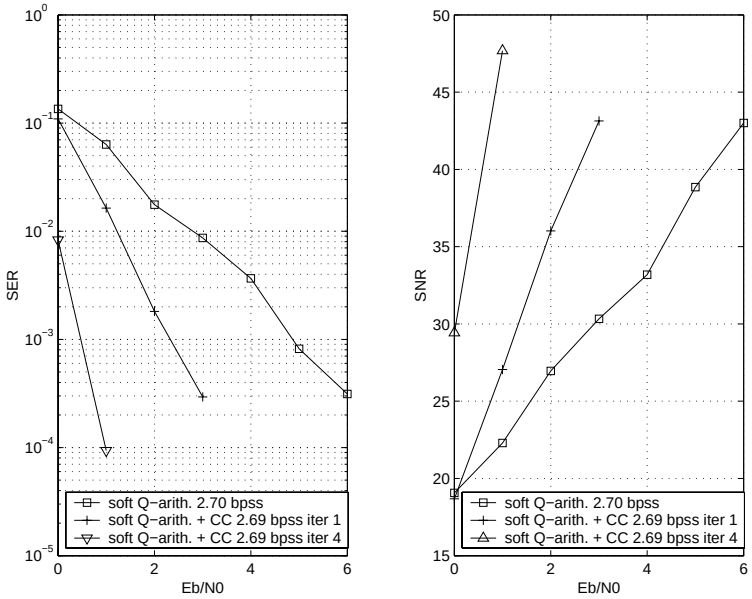


FIG. 4.19 – *SER et SNR pour (a) le décodage souple de codes quasi-arithmétiques et (b) le décodage turbo itératif de codes quasi-arithmétiques ($\rho = 0.9$, 200 symboles, 400 réalisations).*

sion est exploité pour ajouter des marqueurs de synchronisation, ce codeur parvient à rattraper le codeur de précision inférieure pour des faibles taux d'erreur.

En section 4.4.2, nous avons souligné l'importance de l'indexation lors de l'étape de conversion d'une source M-aire vers une source binaire. L'attribution des mots de codes binaires aux symboles M-aire a en effet un impact sur la fiabilité de l'estimation. Cet impact a été vérifié expérimentalement. La figure 4.21 présente un exemple de comparaison des performances de l'algorithme d'estimation avec et sans stratégie d'indexation. L'importance de l'indexation est évidente.

Depuis le début de ce chapitre, nous revendiquons une perte en compression introduite par le codage quasi-arithmétique qui reste raisonnable. Le tableau 4.3 présente les mesures expérimentales de débit excédent pour la source de Gauss-Markov utilisée dans cette section et pour $T = 4$. Pour des sources faiblement corrélées ($\rho \leq 0.5$), le débit excédent est effectivement raisonnable, compte tenu du gain apporté en robustesse. Pour $\rho = 0.9$, en revanche, le débit excédent semble prohibitif. Cela explique les meilleures performances du décodage souple de codes arithmétiques observées au début de cette section. Il ne faut pas oublier cependant que le codage quasi-arithmétique reste plus flexible que les codes de Huffman pour exploiter la statistique de la source. L'arbre de Huffman que nous utilisons dans nos expériences est construit à partir des probabilités stationnaires de la source, et conduit par conséquent à une compression moins efficace que le codage quasi-arithmétique, même pour $\rho = 0.9$. Il est possible d'exploiter la corrélation de la source avec les codes de Huffman, par exemple en créant

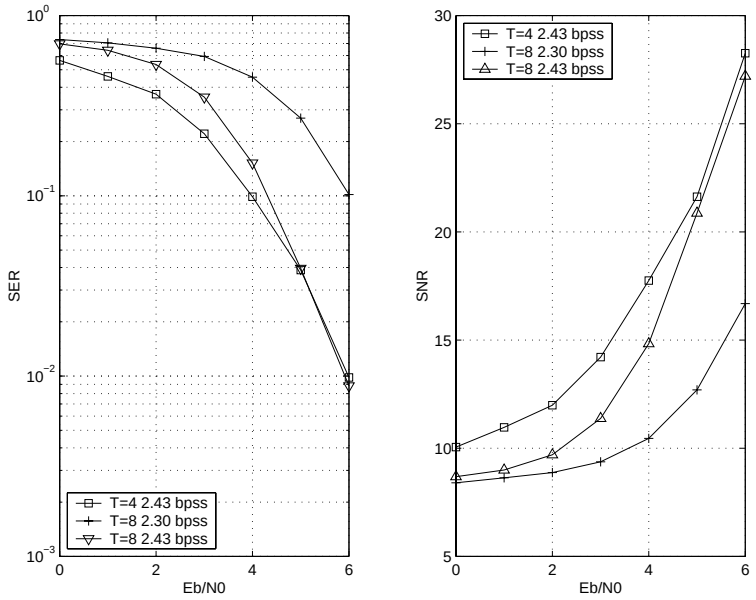


FIG. 4.20 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques avec (a)* $T = 4$ *sans redondance, (b)* $T = 8$ *sans redondance et (c)* $T = 8$ *avec des marqueurs de synchronisation (*$\rho = 0.5$*,* 100 *symboles,* 400 *réalisations).*

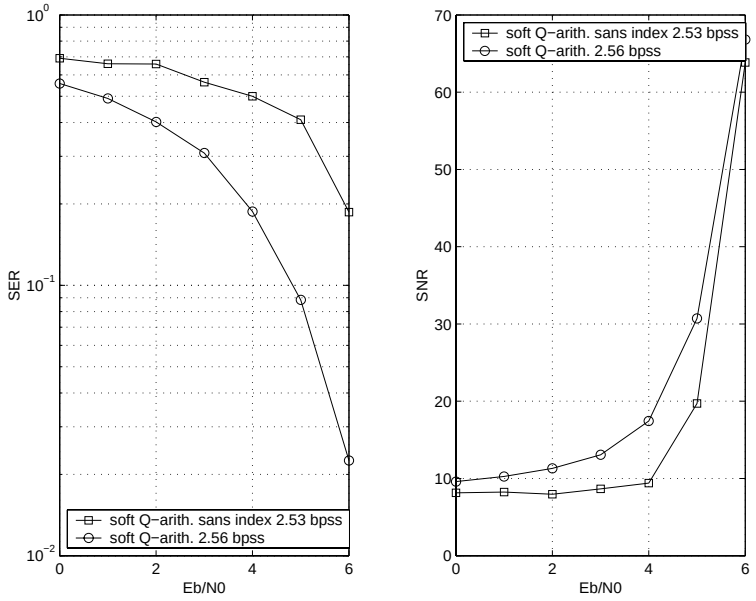


FIG. 4.21 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques (a) sans indexation et (b) avec indexation (*$\rho = 0.1$*,* 200 *symboles,* 400 *réalisations).*

plusieurs arbres de Huffman et en les choisissant en fonction du dernier symbole traité au cours du codage et du décodage. Cependant, cette solution ne serait pas plus efficace que les codes quasi-arithmétiques, avec l'inconvénient d'être moins flexible. Notons, d'ailleurs, que les codes quasi-arithmétiques permettent de compresser des sources binaires, contrairement aux codes de Huffman. Le passage à la précision supérieure $T = 8$ (tableau 4.4) permet de réduire considérablement le débit excédent. Le codeur quasi-arithmétique tend vers les performances du codeur arithmétique avec l'augmentation de T.

ρ	entropie H	débit observé R	débit excédent E (%)
0.1	2.44 bpss	2.53 bpss	0.088 bpss (3.59 %)
0.5	2.24 bpss	2.43 bpss	0.18 bpss (8.19 %)
0.9	1.26 bpss	1.92 bpss	0.67 bpss (53.10 %)

TAB. 4.3 – *Débit excédent d'un codeur quasi-arithmétique mesuré expérimentalement pour* $T = 4$.

ρ	entropie H	débit observé R	débit excédent E (%)
0.1	2.44 bpss	2.46 bpss	0.018 bpss (0.75 %)
0.5	2.24 bpss	2.29 bpss	0.053 bpss (2.35 %)
0.9	1.26 bpss	1.53 bpss	0.27 bpss (21.49 %)

TAB. 4.4 – *Débit excédent d'un codeur quasi-arithmétique mesuré expérimentalement pour* $T = 8$.

Nous revendiquons également dans ce chapitre l'optimalité de l'algorithme d'estimation, dans le sens où le résultat de cette estimation est la séquence la plus vraisemblable parmi *toutes les séquences possibles*. Par opposition, l'algorithme d'estimation du chapitre précédent n'est pas optimal, dans le sens où il a recourt des techniques d'élagage. L'élagage permet de contrôler la complexité du décodage. Dans le cas du décodage souple optimal, la complexité dépend de la taille du treillis d'estimation. Pour une séquence de 200 symboles quantifiés sur 3 bits et pour une précision de $T = 4$, nous observons dans nos expériences un nombre d'états moyen du treillis par bit de la séquence estimée d'environ 2300. Pour une séquence de 50 symboles, ce nombre est d'environ 500, mais si on passe à une précision de $T = 8$, il dépasse de nouveau les 2000. La complexité de l'algorithme d'estimation est donc loin d'être négligeable. Pour certaines applications, il peut être nécessaire de recourir à nouveau à l'élagage, afin de réduire cette complexité. On peut s'interroger dans ce cas sur l'intérêt du codage quasi-arithmétique par rapport au codage arithmétique du chapitre précédent.

Dans une dernière série d'expérience, nous essayons d'analyser la localisation des erreurs dans les séquences estimées. La figure 4.22 est représentative de ce que nous avons observé. Dans le cas du décodage souple de codes arithmétiques, on remarque que le nombre d'erreurs moyen est plus important à la fin qu'au début de la séquence, et ce malgré la présence de marqueurs de synchronisation et la connaissance des deux

variables K et N qui permet de garantir la synchronisation à la fin de la séquence. Ce phénomène est lié d'une part à l'élagage qui rend l'estimation sous optimale et d'autre part à la structure d'arbre du modèle de codeur arithmétique qui empêche la resynchronisation après une erreur. Ainsi, l'algorithme d'estimation tend à éviter des désynchronisation, mais si une erreur se produit malgré tout, l'algorithme est inefficace pour tous les symboles qui suivent cette erreur. Pour le codage quasi-arithmétique, en revanche, on observe une nette diminution du nombre d'erreurs moyen autour des marqueurs de synchronisation et à la fin de la séquence. Grâce à la structure en treillis du modèle produit source + décodeur quasi-arithmétique, il est possible de resynchroniser le décodeur après une erreur, ce qui constitue un avantage important par rapport à l'utilisation des codes arithmétiques, même en cas d'utilisation de techniques d'élagage.

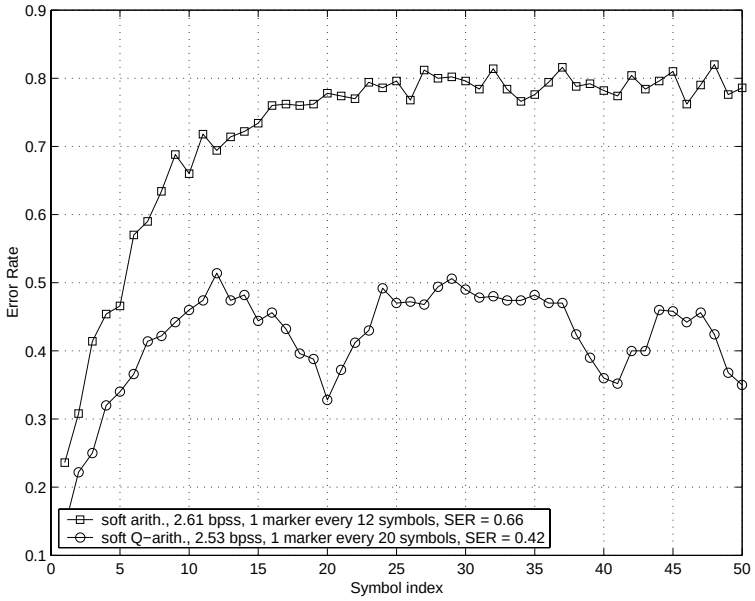


FIG. 4.22 – *Nombre moyen d'erreurs symboles par index dans la séquence ($\rho = 0.5$, 50 symboles, 500 réalisations).*

4.8 Conclusion

Le remplacement du codage arithmétique par le codage quasi-arithmétique permet d'obtenir un nouveau compromis compression/robustesse. Pour une perte minime en compression, il est possible de concevoir un algorithme de décodage souple qui ne nécessite pas le recours à l'élagage. Là encore, les réseaux bayésiens offrent un cadre adapté à l'analyse des structures de dépendances entre les variables impliquées dans le processus de codage et de décodage. Pour une précision suffisament réduite, le codeur quasi-arithmétique est modélisé par un automate à nombre d'états finis. Comme un tel codeur est plus efficace pour compresser des sources binaires, nous incluons une étape

de conversion de source M-aire vers binaire dans la chaîne de codage. Finalement, un modèle produit de l'ensemble des éléments de la chaîne de codage est obtenu. Il s'agit également d'un automate à nombre d'états fini. Un algorithme d'estimation optimal se déduit naturellement. La redondance du train binaire peut être augmentée en incluant soit des marqueurs de synchronisation, soit de l'information adjacente dans le processus de codage. Cette redondance est exploitée par l'algorithme d'estimation et permet d'augmenter la robustesse du schéma de codage. La concaténation d'un code convolutif pour la conception d'un algorithme de décodage itératif inspiré des turbo codes en série a également été considérée. Contrairement au cas du chapitre précédent, l'absence d'élagage permet aux itérations d'apporter un gain significatif.

Les résultats présentés ont effectivement été calculés sur des treillis d'estimation "complets". Il est important de signaler, cependant, que de part la nature de codes à longueur variable des codes quasi-arithmétiques, qui est prise en compte grâce au compteur de symboles K_n, les treillis d'estimation peuvent atteindre des tailles importantes, en particulier pour de longues séquences. Si cette taille devient trop importante pour l'application visée, il redevient nécessaire de recourir à des techniques d'élagage. Dans ce cas, les codes quasi-arithmétiques présentent tout de même un avantage par rapport aux codes arithmétiques de part leur meilleure capacité de resynchronisation. En effet, comme cela est illustré par la figure 4.22, la structure d'arbre des codes arithmétiques empêche toute resynchronisation, alors que la structure en treillis des codes quasi-arithmétiques permet cette resynchronisation.

Enfin pour terminer, nous pouvons faire des remarques d'ordre général sur les résultats. Tout d'abord, pour une quantité de redondance donnée, le choix du mode d'ajout de cette redondance dépend de la nature de la source. En effet, pour une source faiblement corrélée, il est plus efficace d'augmenter directement cette corrélation plutôt que d'ajouter une redondance complémentaire, par exemple sous forme de codes de canal. Les sources fortement corrélées, en revanche, contiennent suffisament d'information exploitable par le décodeur. L'augmentation de cette corrélation se révèle alors inefficace. Il est préférable d'utiliser des codes de canal. Nous pouvons signaler également l'importance de l'aspect compression dans la chaîne de codage. S'il est parfois dit, dans le cadre du codage conjoint source/canal, qu'il est préférable de compresser moins pour atteindre une meilleure robustesse, nos résultats montrent au contraire qu'il ne faut pas négliger la compression. Plus précisément, comme cela est illustré par exemple sur la figure 4.14, où le décodage souple optimal de codes quasi-arithmétiques est surpassé par le décodage souple non optimal de codes arithmétiques, une meilleure compression laisse de la place qui, si elle est exploitée de façon appropriée, permet de traiter *spécifiquement* la fragilité des codes utilisés.

Dans ces deux derniers chapitres, nous avons considéré le problème de la transmission de données compressées sur des canaux à erreurs bits. Plus précisément, nous nous sommes concentré sur la synchronisation des codes arithmétiques, qui sont reconnus comme étant le point faible d'un système de compression en cas d'erreurs bits. Les algorithmes de décodage souple proposés permettent d'atteindre des compromis compression/robustesse particulièrement intéressants. Nous considérons à présent le problème de l'utilisation de ces techniques conjointement avec le codage par descrip-

tions multiples, dans le but de concevoir un schéma de codage résistant à la fois aux erreurs et aux effacements.

Chapitre 5

Décodage souple de descriptions multiples

5.1 Introduction

Dans les chapitres 1 et 2, nous avons fait un état de l'art du codage par descriptions multiples et nous avons proposé une application au codage d'image fixe progressif basé JPEG2000. Le codage par descriptions multiples se révèle être un outil performant pour garantir la fiabilité d'une transmission de données sur un canal à effacements. Cependant, il est généralement utilisé sous l'hypothèse d'une erreur bit résiduelle nulle. Cette hypothèse est réaliste dans certains cas, comme par exemple dans celui de la transmission sur le réseau internet filaire. Actuellement, les réseaux deviennent de plus en plus hétérogènes et peuvent inclure des transmissions sans fil. Dans ce dernier cas, de nombreuses perturbations peuvent intervenir et il devient impossible de garantir le respect de l'hypothèse d'erreur bit résiduelle nulle. Il a été signalé en section 1.5 que le codage par descriptions multiples n'est pas adapté au traitement des erreurs bits. Il convient donc de le coupler à un autre mécanisme de protection.

Dans les chapitres 3 et 4, nous avons mis en avant le point faible d'une chaîne de codage classique lorsqu'elle est confrontée à des erreurs bits, à savoir le codage arithmétique. En effet, une seule erreur bit suffit à désynchroniser un décodeur arithmétique. Tous les symboles qui suivent l'erreur sont alors erronés. Ce problème de synchronisation a été traité par des algorithmes de décodage souple et des modes d'ajout de redondance adaptés. Le remplacement des codes arithmétiques traditionnels par les codes quasi-arithmétiques a permis d'atteindre de meilleurs compromis compression/robustesse.

Nous cherchons à présent à concevoir un schéma de codage résistant à la fois aux erreurs et aux effacements. Pour cela, il ne suffit pas de concaténer directement le codage par descriptions multiples avec le décodage souple de codes arithmétiques. En effet, chacun de ces deux éléments introduit sa propre redondance. Par conséquent, une simple concaténation risque d'aboutir à un surplus de redondance qui ne sera pas exploité de manière optimale. Idéalement, les deux éléments devraient cohabiter et s'enrichir mutuellement. Ainsi la redondance introduite par le codage à descriptions multiples

pourrait être exploitée par l'algorithme de décodage souple. De même, l'information souple résultante de l'estimation pourrait être exploitée lors de la reconstruction de la source à partir des descriptions reçues.

Dans ce chapitre, nous étudions deux possibilités. Tout d'abord, une chaîne de codage composée d'une quantification scalaire uniforme à descriptions multiples (MDUSQ) et d'un codeur de Huffman est modélisée sous forme de réseau bayésien. L'algorithme d'estimation qui en découle exploite à la fois la corrélation de la source et la corrélation entre les deux descriptions. La MDUSQ constitue alors le seul ajout de redondance. Ensuite, nous considérons une chaîne de codage incluant une MDUSQ, une conversion de source M-aire vers binaire et un codeur quasi-arithmétique. Encore une fois, nous modélisons cette chaîne par un réseau bayésien. Cependant, les contraintes de ce schéma rendent la conception d'un algorithme d'estimation délicate. Par conséquent plusieurs options sont proposées. Finalement l'intérêt de ce schéma est mis en évidence par des expériences dans un contexte de transmission avec effacements et erreurs.

5.2 Notations

Nous modifions les notations utilisées au chapitre précédent pour les étendre au cas du codage à deux descriptions. Soit $A = A_1 \dots A_L$ une séquence de symboles de source quantifiés qui prennent leur valeur dans un alphabet fini $\mathcal{A} = \{a_1, a_2, \dots, a_r, \dots, a_R\}$ composé de R symboles. Nous supposons que la séquence $A = A_1 \dots A_L$ est une chaîne de Markov d'ordre 1. Cette séquence de symboles M-aires est convertie en deux descriptions $I = I_1 \dots I_L$ et $I' = I'_1 \dots I'_L$ par une table d'index MDUSQ. Les symboles de ces deux séquences prennent leurs valeurs respectivement dans les alphabets $\mathcal{I} = \{i_1, i_2, \dots, i_m, \dots, i_M\}$ et $\mathcal{I}' = \{i'_1, i'_2, \dots, i'_m, \dots, i'_M\}$, composés de $M = 2^q$ symboles. Si cela est nécessaire, ces deux descriptions sont converties indépendamment en deux séquences de symboles binaires $S = S_1 \dots S_K$ et $S' = S'_1 \dots S'_K$, avec $K = q \times L$. Les deux sources, M-aires ou binaires, sont converties à leur tour en deux séquences de bits d'information $U = U_1 \dots U_N$ et $U' = U'_1 \dots U'_{N'}$, par l'intermédiaire soit d'un codeur de Huffman, soit d'un codeur arithmétique, soit d'un codeur quasi-arithmétique de précision T. Les longueurs N et N' de ces deux trains binaires sont des variables aléatoires, fonctions de la séquence de symboles M-aires originale A. Les deux trains binaires U et U' sont transmis sur un canal sans mémoire et reçus comme des mesures Y et Y'. Le problème que nous considérons consiste à estimer A, connaissant soit les observations sur chacune des descriptions y et y', soit les observations sur une seule des descriptions y ou y'. Nous utilisons les lettres majuscules pour dénoter les variables aléatoires, et les minuscules pour les réalisations de ces variables. Pour désigner une suite de variables successives, nous utilisons la notation $X_{b_1}^{b_2} = \{X_{b_1}, X_{b_1+1}, \dots, X_{b_2}\}$ ou $\bar{X}_B$ avec B l'ensemble des index $\{b_1, b_1 + 1, \dots, b_2\}$.

5.3 Descriptions multiples et codes de Huffman

Nous nous intéressons dans un premier temps à la combinaison du codage par descriptions multiples et des codes de Huffman. Nous désignons par *codes MDUSQ+VLC* le schéma de codage ainsi proposé. Nous proposons un algorithme de décodage souple inspiré du travail de Guyader, Fabre, Guillemot et Robert [GFGR01] (voir également annexe B).

5.3.1 Modélisation

5.3.1.1 dépendances entre les descriptions

Nous supposons que la source $A = A_1 \dots A_L$ est une chaîne de Markov d'ordre 1, dont nous connaissons les distributions stationnaires $\mathbb{P}(A_l)$ et les probabilités de transitions $\mathbb{P}(A_l|A_{l-1})$. La séquence de symboles A est convertie en deux séquences d'index $I = I_1 \dots I_L$ et $I' = I'_1 \dots I'_L$ par une table d'index MDUSQ emboîtée (voir section 2.2.3.2). La table d'index définit non seulement la correspondance entre la séquence A et la paire de séquences (I, I'), mais aussi la correspondance entre les distributions de ces séquences. Ainsi la distribution stationnaire de la description I est donnée par

$$\mathbb{P}(i_m) = \sum_{r:ia(a_r)=i_m} \mathbb{P}(a_r),\ m = 1, 2, \dots M, \tag{5.1}$$

avec $ia(a_r)$ le symbole i_m correspondant à a_r, donné par la table d'index MDUSQ pour la description I. De même, la distribution conditionnelle de la description I est donnée par

$$\mathbb{P}(i_m|i_p) = \sum_{r:ia(a_r)=i_m} \sum_{s:ia(a_s)=i_p} \mathbb{P}(a_r|a_s),\ m, p = 1, 2, \dots M. \tag{5.2}$$

Les deux descriptions I et I' sont également des chaînes de Markov d'ordre 1. Ces deux descriptions sont codées en deux séquences de bits d'information U et U' par un codeur de Huffman. Ces deux trains binaires sont transmis sur un canal sans mémoire et reçus sous la forme d'observations Y et Y'.

5.3.1.2 Modèle de codeur de Huffman

La difficulté dans l'estimation de la source A sachant les observations y et y' provient de la désynchronisation entre les états cachés représentant la source et les mesures bruitées sur les séquences de bits utiles. En d'autres termes, la segmentation des deux trains binaires U et U' en mots de codes est aléatoire. Nous montrons par la suite que l'estimation peut être effectuée en trois étapes. Tout d'abord, les deux trains binaires U et U' sont estimés indépendamment en utilisant la redondance résiduelle entre les mots de codes et en supposant que les symboles A_l sont indépendants. le résultat de cette estimation prend la forme d'information souple sur les états du modèle à horloge bit du codeur de Huffman. Cette information doit être convertie en information souple à

horloge symbole. On peut alors dans un second temps fusionner l'information résultant de l'estimation des deux descriptions. Finalement, la séquence de symboles est estimée à son tour en utilisant la corrélation entre les symboles.

Le modèle de dépendances entre les symboles provient directement de l'hypothèse que A est une chaîne de Markov. Pour générer le modèle de dépendance entre les mots de codes, le codeur de Huffman peut être modélisé par un automate stochastique, où les variables d'états X_n désignent les noeuds de l'arbre de Huffman. Cependant, dans le cas des codes à longueur variable, la connaissance de l'index à horloge bit n ne suffit pas pour déterminer l'index à horloge symbole l du symbole en cours de reconstruction. Cette information est nécessaire pour convertir l'information souple sur les bits en information souple sur les symboles. Par conséquent, cet index doit être disponible conjointement avec la variable d'état X_n. Nous augmentons donc X_n avec un compteur L_n du nombre de symboles décodés à l'instant n. Cela revient à définir la distribution des paires (X_n, L_n) comme une chaîne de Markov. La probabilité de transition de (X_n, L_n) vers (X_{n+1}, L_{n+1}) est déterminée par $\mathbb{P}(U_{n+1}|U_n)$, qui elle même dépend de $\mathbb{P}(\bar{U}_{L_n+1}|\bar{U}_{L_n})$. Pour la partie compteur, on a $L_{n+1} = L_n + 1$ chaque fois que X_{n+1} atteint une feuille de l'arbre de Huffman et $L_{n+1} = L_n$ sinon. De la même manière, la chaîne de Markov $A = A_1 \dots A_L$ est augmentée de deux compteurs de bits N_l et N'_l correspondant aux séquences de bits générées par le codage de Huffman des deux descriptions I et I'. Cette modélisation est détaillée dans [GFGR01]. Les dépendances entre les variables de ce modèle sont représentées graphiquement par la figure 5.1.

5.3.2 Estimation

Pour pouvoir effectuer l'estimation de la séquence de symboles, il faut tout d'abord obtenir $\mathbb{P}(\bar{Y}_l, \bar{Y}'_l|A_l, N_l, N'_l)$ qui servira d'entrée à un algorithme d'estimation en deux passes basé sur le modèle de source. Étant donnée la structure d'arbre des dépendances, les observations sur chacun des canaux sont indépendantes conditionnellement aux deux séquences d'index I et I'. Par conséquent, cette information souple est obtenue en fusionnant l'information souple sur les deux processus (I, N) et (I', N') :

$$\mathbb{P}(\bar{Y}_l, \bar{Y}'_l|A_l, N_l, N'_l) = \mathbb{P}(\bar{Y}_l|I_l, N_l)\mathbb{P}(\bar{Y}'_l|I'_l, N'_l)|_{A_l=(I_l,I'_l)}. \tag{5.3}$$

Notons que les deux processus (I, N) et (I', N') sont supposés indépendants conditionnellement aux observations Y et Y'. Il a été montré dans [GFGR01] que

$$\mathbb{P}(\bar{Y}_l|I_l, N_l) \propto \frac{\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^l)}{\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^{l-1})}, \tag{5.4}$$

où $\dot{\mathbb{P}}(I_l, N_l|\bar{Y}_1^l)$ est défini par $\dot{\mathbb{P}}(I_l, N_l|\bar{Y}_1^l) \propto \mathbb{P}(\bar{Y}_1^l = \bar{y}_1^l, I_l, N_l)$ et représente la quantité $\mathbb{P}(I_l, N_l|\bar{Y}_1^l)$ *pour une valeur particulière* y des observations, $Y = y$ (de même pour $\mathbb{P}(I'_l, N'_l|\bar{Y'}_1^l)$). La notation $\propto$ désigne une renormalisation sur (I_l, N_l). L'indice i dénote les distributions de probabilité sous l'hypothèse de l'indépendance des symboles. Il est nécessaire de calculer tout d'abord l'information souple sur les processus (I_l, N_l) et (I'_l, N'_l), $l = 1, \dots, L$. Ces deux processus peuvent être estimés séparément.

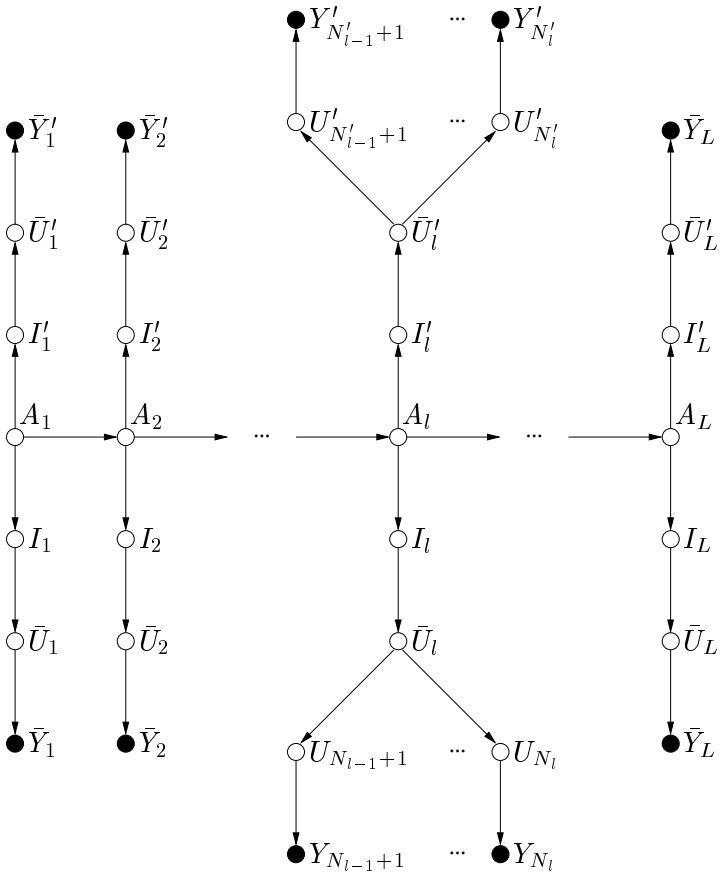


FIG. 5.1 – *Structure de dépendances du modèle source de Markov + MDUSQ + codeur de Huffman. Les états blancs et noirs représentent respectivement les variables cachées et observées.*

Considérons à présent le calcul de $\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^l)$. Certaines valeurs de (X_n, L_n) correspondent aux feuilles de l'arbre de Huffman, et donc aux symboles possibles à l'instant $l = L_n$. Les probabilités $\mathbb{P}_i(X_n, L_n|Y)$ correspondent alors aux probabilités $\mathbb{P}_i(I_l, N_l|Y)$ recherchées. Par conséquent, pour obtenir l'information souple sur les processus (I, N), il suffit d'estimer les processus augmentés (X, L). Sous l'hypothèse de l'indépendance des symboles, la quantité $\mathbb{P}_i(X_n, L_n|Y)$ s'obtient en une seule passe d'estimation MPM sur le modèle du codeur de Huffman. En effet, l'indépendance des symboles entraîne $\mathbb{P}_i(X_n, L_n|Y) = \mathbb{P}_i(X_n, L_n|Y_1^n)$. Par conséquent, la quantité $\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^l)$ peut être calculée par une passe avant d'estimation sur le modèle de codeur de Huffman, si on suppose les symboles indépendants. La quantité $\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^{l-1})$ est donnée par

$$\dot{\mathbb{P}}_i(I_l, N_l|\bar{Y}_1^{l-1}) = \mathbb{P}(I_l) \cdot \dot{\mathbb{P}}_i(N_{l-1}|\bar{Y}_1^{l-1})\Big|_{N_{l-1}=N_l-\mathcal{L}(I_l)}, \tag{5.5}$$

où $\mathcal{L}(I_l)$ représente la longueur du mot de code associé à l'index I_l.

L'information souple $\mathbb{P}(A_l, N_l, N_l'|\bar{Y}_l, \bar{Y}_l')$ obtenue en fusionnant l'information souple sur chacune des deux descriptions (équation 5.3) est utilisée comme entrée par une procédure qui va estimer le processus (A, N, N') en deux passes. La difficulté de cette estimation réside dans la structure aléatoire du réseau bayésien qui relie les différents modèles. Cependant, sachant le processus A, et par conséquent I, I', N et N', la structure de l'arbre est fixée : Pour toute valeur particulière (a_l, n_l, n_l') du triplet (A_l, N_l, N_l'), les vecteurs $\bar{Y}_1^l = Y_1 \dots Y_{N_l}$, $\bar{Y}_{l+1}^L = Y_{N_l+1} \dots Y_N$, $\bar{Y'}_1^l = Y_1' \dots Y_{N_l'}'$ et $\bar{Y'}_{l+1}^L = Y_{N_l'+1}' \dots Y_{N'}'$ sont parfaitement définis. Si on définit $\dot{\mathbb{P}}(A_l, N_l, N_l'|\bar{Y}_1^l, \bar{Y'}_1^l)$ pour une valeur donnée des observations y et y' par $\dot{\mathbb{P}}(A_l, N_l, N_l'|\bar{Y}_1^l, \bar{Y'}_1^l) \propto \mathbb{P}(\bar{Y}_1^l = \bar{y}_1^l, \bar{Y'}_1^l = \bar{y'}_1^l, A_l, N_l, N_l')$, alors l'équation

$$\mathbb{P}(A_l, N_l, N_l'|Y, Y') \propto \dot{\mathbb{P}}(A_l, N_l, N_l'|\bar{Y}_1^l, \bar{Y'}_1^l) \cdot \dot{\mathbb{P}}(\bar{Y}_{l+1}^L, \bar{Y'}_{l+1}^L|A_l, N_l, N_l')$$

est valide pour ces valeurs particulières de y et y' et constitue une décomposition qui permet d'effectuer l'estimation de (A, N, N').

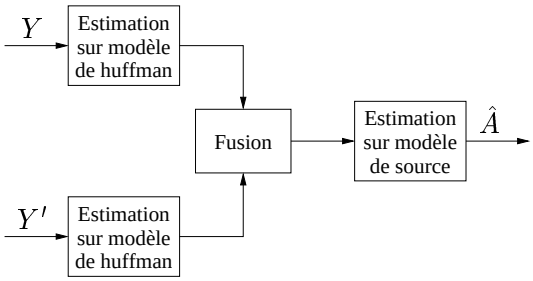


FIG. 5.2 – *Décodage souple de MDUSQ et de codes de Huffman.*

Finalement, l'estimation se décompose en trois étapes résumées sur la figure 5.2. La première étape se base sur les modèles d'arbres de Huffman pour calculer $\mathbb{P}(\bar{Y}_l|I_l, N_l)$ et $\mathbb{P}(\bar{Y}_l'|I_l', N_l')$ indépendamment sur chacune des descriptions. La seconde étape fusionne

l'information souple résultante des deux descriptions afin d'obtenir $\mathbb{P}(\bar{Y}_l, \bar{Y}_l'|A_l, N_l, N_l')$. Enfin, la troisième étape utilise le modèle de source pour calculer $\mathbb{P}(A_l, N_l, N_l'|Y, Y')$. Ces lois a posteriori permettent alors d'obtenir la séquence la plus vraisemblable au sens du MAP.

5.3.3 Résultats expérimentaux

Afin d'évaluer les performances du schéma de décodage souple de codes MDUSQ+ VLC, une série d'expériences a été réalisée sur une source de Gauss-Markov d'ordre 1 de moyenne nulle et de variance unité avec différents facteurs de corrélation ρ. Le débit est de 3 bits par échantillon par description et la largeur de la diagonale de la table d'index MDUSQ est $d = 2$. La source est donc quantifiée sur 34 niveaux uniformes, ce qui est proche de 5 bits par symbole. La redondance introduite par la MDUSQ est approximativement équivalente à un rendement 5/6, suivant la terminologie du codage de canal. Nous considérons des séquences de longueur $K = 50$ symboles. Le codage VLC est basé sur deux codes de Huffman adaptés aux distributions stationnaires de chacune des deux descriptions. Toutes les simulations ont été effectuées pour un canal gaussien à bruit blanc additif (AWGN) avec une modulation BPSK. Les résultats sont moyennés sur 500 réalisations.

Les figures 5.3, 5.4 et 5.5 présentent les courbes de SER et SNR résiduels en fonction du rapport signal à bruit E_b/N_0 du canal, respectivement pour $\rho = 0.1$, $\rho = 0.5$ et $\rho = 0.9$. Sur chaque graphique, le décodage souple de codes MDUSQ+VLC est comparé au décodage souple de codes de Huffman sans redondance ajoutée et au décodage conjoint turbo source/canal de codes VLC avec un code convolutif systématique récursif de rendement 1/2 poinçonné à 5/6 (4 itérations). Les débits obtenus sont équivalents. Les performances relatives des différentes approches dépendent de la corrélation présente dans la source et de la fiabilité du canal. Pour des sources faiblement corrélées ($\rho \leq 0.5$), et pour $E_b/N_0 \leq 4$ dB, la MDUSQ apporte un gain de 2 à 4 dB en terme de SNR par rapport au décodage turbo. En revanche, pour des sources fortement corrélées, le décodage turbo surpasse la MDUSQ. Cette observation a déjà été faite dans les chapitres précédents : lorsque la corrélation de la source est faible, il est plus efficace de rajouter de la redondance directement dans cette source. En revanche, lorsque la corrélation est forte, l'ajout de redondance supplémentaire n'apporte que peu d'information. Il est alors plus efficace d'utiliser un moyen de protection séparé, tel qu'un code de canal.

Des expériences ont également été réalisées en variant le niveau de redondance. Elles montrent que la MDUSQ surpasse le décodage turbo pour de faibles niveaux de redondance ($d \geq 2$).

5.4 Descriptions multiples et codes quasi-arithmétiques

Dans la première partie de ce chapitre, nous avons adapté l'algorithme de décodage souple de codes de Huffman proposé dans [GFGR01] au codage par descriptions multiples. Le concept de décodage souple de descriptions multiples ainsi développé doit

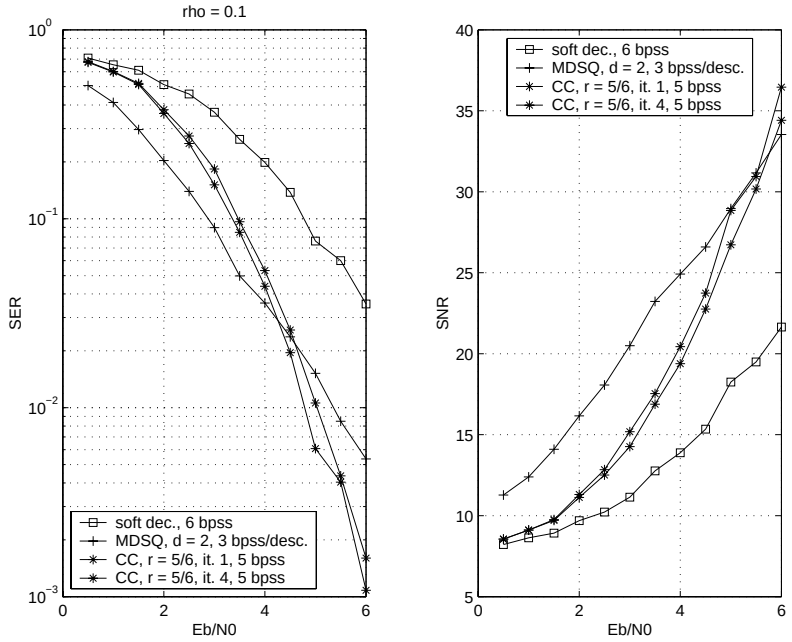


FIG. 5.3 – *SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement* $5/6$*).* $\rho = 0.1$.

permettre de lutter à la fois contre les erreurs et les effacements (ce point est validé expérimentalement à la fin du chapitre). Nous nous intéressons à présent, dans une deuxième partie, au décodage souple de codes arithmétiques. La combinaison des algorithmes d'estimation du chapitre 3 avec la MDUSQ est relativement aisée. Cependant, un tel schéma est limité par la sous-optimalité de l'estimation. Nous préferons mettre l'accent sur l'utilisation du codage quasi-arithmétique. Nous désignons par *codes MDUSQ+QAC* le schéma de codage proposé.

Le codage quasi-arithmétique nécessite un traitement particulier pour être combiné au codage par descriptions multiples. L'algorithme d'estimation proposé dans la première partie de ce chapitre ne peut pas être utilisé. Tout d'abord, il faut inclure une étape de conversion de source M-aire vers source binaire dans le modèle de dépendances. En effet, nous avons vu dans le chapitre 4 que cette conversion apportait une meilleure efficacité de compression pour une complexité réduite. Par ailleurs, la MDUSQ ne peut pas s'appliquer sur des sources binaires. Cela nous conduit à considérer uniquement des applications où les séquences de symboles prennent leurs valeurs dans des alphabets M-aire, avec M suffisament grand. Si cette contrainte n'est pas respectée, il devient indispensable d'envisager une autre technique de codage par descriptions multiples que la MDUSQ. Ensuite, il est préférable de proposer un algorithme qui n'a pas besoin d'effectuer de conversion d'horloge. En effet, si une telle conversion est aisée dans le cas des codes de Huffman, cela n'est pas le cas pour les codes quasi-arithmétiques, car les

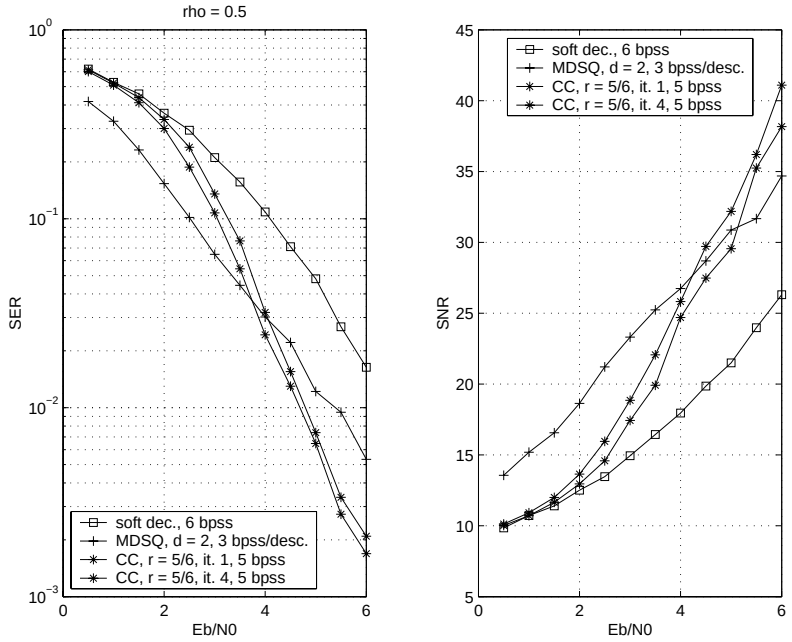


FIG. 5.4 – *SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement* $5/6$*).* $\rho = 0.5$.

modèles de codeurs et de décodeurs ne sont pas synchronisés. De plus, nous avons vu qu'il était préférable d'effectuer l'estimation sur le modèle de décodeur à horloge bit, car le nombre d'état du modèle de codeur n'est pas borné (sauf pour la précision minimale $T = 4$). La difficulté supplémentaire imposée par ces contraintes nous conduisent à proposer plusieurs algorithmes d'estimation, dont certains ne sont pas optimaux, faute d'exploiter la totalité des dépendances présentes entre les variables de la chaîne de codage.

5.4.1 Modélisation

De la même manière qu'en section 5.3.1.1, la séquence de symboles $A = A_1 \dots A_L$ est convertie en deux séquences d'index $I = I_1 \dots I_L$ et $I' = I'_1 \dots I'_L$ par une table d'index MDUSQ emboîtée. Ces deux descriptions sont ensuite converties en deux séquences de symboles binaires $S = S_1 \dots S_K$ et $S' = S'_1 \dots S'_K$, avec $K = q \times L$. Les détails de cette conversion sont donnés en section 4.4. Un modèle de source binaire pour chacune des deux descriptions est obtenu durant cette conversion. Ces modèles de sources sont associés à un modèle de codeur quasi-arithmétique pour former deux modèles produits, tel que cela est décrit en section 4.5. Les symboles binaires S_k (respectivement S'_k) déclenchent les transitions entre les états X_{k-1} et X_k (respectivement X'_{k-1} et X'_k) du modèle produit. Finalement, les deux codeurs quasi-arithmétiques ainsi définis permettent de convertir les deux séquences de symboles binaires S et S' en deux trains

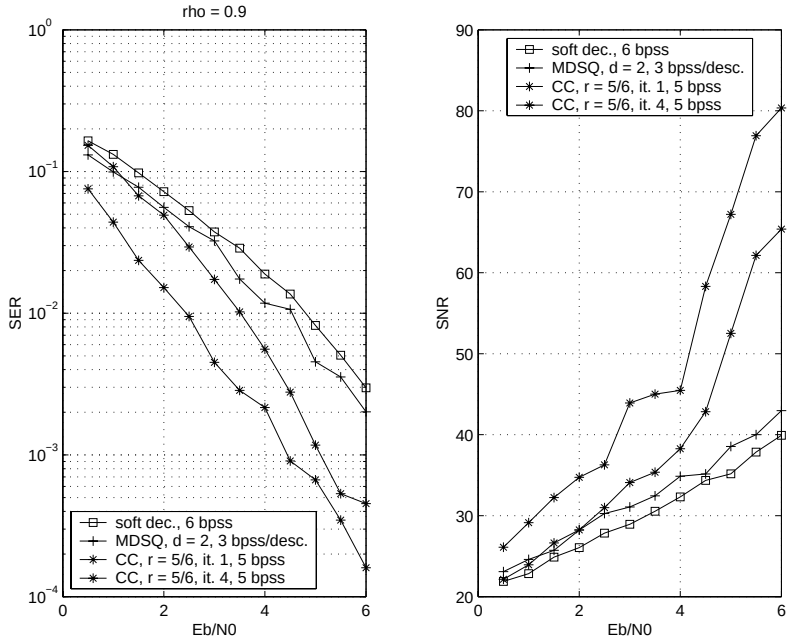


FIG. 5.5 – *SER et SNR pour (a) le décodage souple de codes VLC, (b) le décodage souple de codes MDUSQ+VLC ($d = 2$) et (c) le décodage conjoint turbo source/canal de codes VLC (rendement* $5/6$*).* $\rho = 0.9$.

de bits d'information $U = U_1 \dots U_N$ et $U' = U'_1 \dots U'_{N'}$. Ces deux trains binaires sont transmis sur un canal sans mémoire ou sur deux canaux indépendants et sont reçus sous la forme de deux séquences d'observations $Y = Y_1 \dots Y_N$ et $Y' = Y'_1 \dots Y'_{N'}$.

Le codeur proposé est résumé schématiquement sur la figure 5.6. Les dépendances entre les variables de ce modèle sont représentées graphiquement sur la figure 5.7, pour une seule des deux descriptions, la seconde description étant symétrique. Contrairement aux codes de Huffman, où les symboles des deux descriptions étaient supposés indépendants, la corrélation entre les symboles binaires est prise en compte par les codeurs quasi-arithmétiques. Elle sera également prise en compte au décodage, lors de l'estimation.

5.4.2 Estimation

Nous proposons trois algorithmes d'estimation. Le premier ne s'applique que dans le cas particulier d'un codeur quasi-arithmétique de précision minimale ($T = 4$). De plus, il peut souffrir d'une complexité élevée. Nous considérons un deuxième algorithme, moins complexe et plus général, mais sous-optimal. Finalement, nous proposons en perspective un troisième schéma d'estimation inspiré du principe des turbo codes.

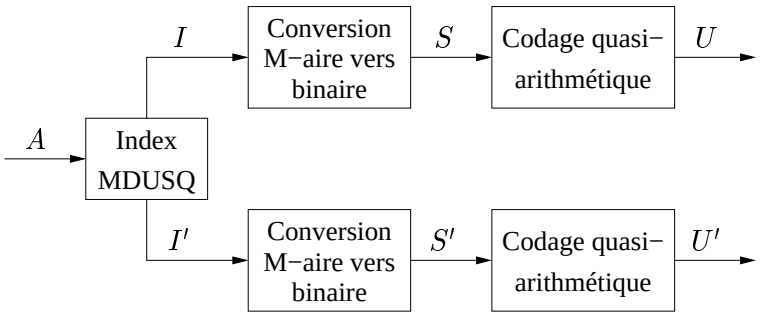


FIG. 5.6 – *Schéma bloc d'un codeur quasi-arithmétique à deux descriptions.*

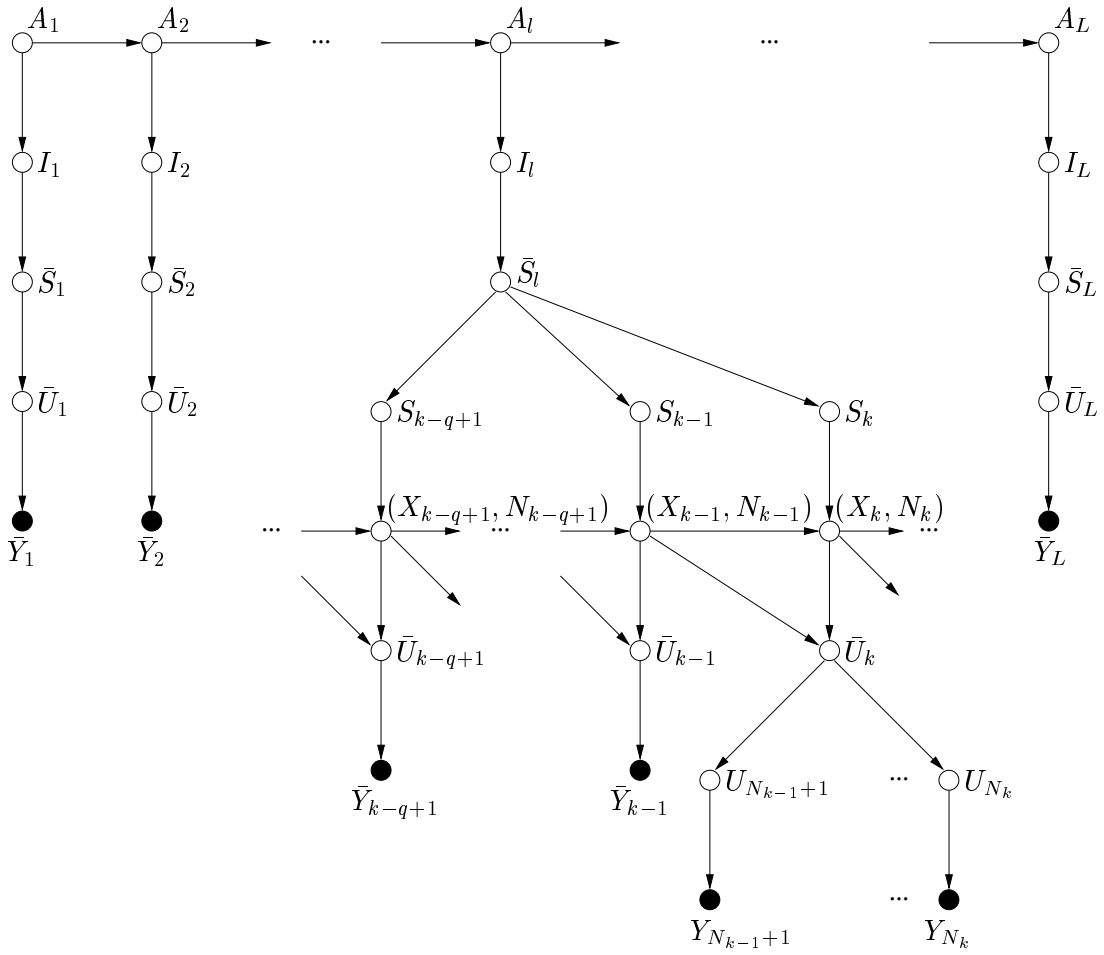


FIG. 5.7 – *Structure de dépendances du modèle source de Markov M-aire + MDUSQ + conversion vers source binaire + codeur quasi-arithmétique, pour une seule description (la deuxième description est symétrique). On a $k = l \times q$. Les états blancs et noirs représentent respectivement les variables cachées et observées.*

5.4.2.1 Estimation basée produit

Dans un premier temps, nous avons cherché à estimer l'ensemble des états cachés $(X, K, X', K') = (X_1, K_1, X'_1, K'_1) \dots (X_N, K_N, X'_{N'}, K'_{N'})$ directement, suivant un algorithme équivalent à celui présenté en section 4.6. L'idée revient à effectuer un produit des deux processus (X, K) et (X', K'), au sens du produit de treillis. Cependant, une telle technique est difficile à justifier et à mettre en oeuvre car elle fonctionne à horloge bit. Or nous devons manipuler deux horloges bits distinctes n et n' qui ne sont pas synchronisées ($N \neq N'$). Dans le cas des codes de Huffman, ce problème est résolu par une estimation séparée des deux descriptions, suivie d'une conversion d'horloge qui permet de synchroniser l'information souple obtenue sur chacune des descriptions. La fusion de l'information qui provient des deux descriptions se fait alors naturellement.

Dans le cas des codes quasi-arithmétiques, la conversion d'horloge est rendue délicate par la disymétrie entre le codeur et le décodeur d'une part et par le fait que le nombre d'états du modèle de codeur n'est pas borné d'autre part. Ce dernier point provient de la variable n_{scl} qui doit nécessairement être incluse dans le modèle du codeur quasi-arithmétique. Cette variable permet de mémoriser des remises à l'échelles de l'intervalle de probabilité qui désigne la séquence de symboles codés, et ce afin d'éviter des problèmes de précision numérique (sections 3.3 et 4.3.1). Cependant, comme on le constate sur les exemples donnés dans les sections 4.3.1 et 4.3.2, dans le cas d'un codeur quasi-arithmétique de précision minimale, c'est à dire pour $T = 4$, la variable n_{scl} est systématiquement égale à 0, et peut donc être ignorée. Le problème de l'estimation de descriptions multiples codées quasi-arithmétiquement peut donc être étudié dans le cas particulier où $T = 4$. Il est également possible d'étudier ce problème d'estimation pour tous les cas où la combinaison du modèle de codeur et du modèle de source conduit à un modèle produit qui comporte un nombre d'états fini. Ne sachant pas déterminer les exemples pour lesquels cette contrainte est respectée autrement qu'expérimentalement, nous préferons nous restreindre au cas où $T = 4$.

Une première façon de traiter le problème consiste alors à adopter la même démarche que pour les codes de Huffman. Cependant, l'étape de conversion d'horloge reste délicate, du fait de la désynchronisation entre le codeur et le décodeur. Dans le cas des codes de Huffman, chaque feuille de l'arbre de codage correspond à un et un seul symbole. Dans le cas du codage quasi-arithmétique, la transition entre deux états (X_{n-1}, K_{n-1}) et (X_n, K_n) du modèle de décodeur à horloge bit est déclenchée par le bit U_n. L'émission des symboles $S_{K_{n-1}+1}^{K_n}$ est associée à cette transition. De même, la transition entre deux états (X_{k-1}, N_{k-1}) et (X_k, N_k) du modèle de codeur à horloge symbole est déclenchée par le symbole S_k. L'émission des bits $U_{N_{k-1}+1}^{N_k}$ est associée à cette transition. Si on devait identifier deux états (X_n, K_n) et (X_k, N_k) de ces deux modèles, alors on souhaiterait respecter les contraintes $K_n = k$, $N_k = n$, $U_{N_k} = U_n$ et $S_{K_n} = S_k$. Cependant, les deux dernières contraintes ne sont pas associées aux états, mais à des transitions entre états. De plus ces transitions peuvent émettre plusieurs symboles ou bits. Les deux premières contraintes sont nécessaires pour identifier les deux états (X_n, K_n) et (X_k, N_k), mais ne sont pas suffisantes. La question de l'existence d'états pouvant être identifiés n'est pas résolue.

Compte tenu de ces difficultés, nous proposons une deuxième façon de traiter le problème, basée sur une formulation à horloge symbole. Cette technique d'estimation reprend à la fois les équations de la section 3.5 et le principe de l'algorithme BCJR utilisé en section 4.6. Le problème consiste à estimer l'ensemble des états cachés $(A_l, N_l, N'_l) = (A_1, N_1, N'_1) \dots (A_L, N_L, N'_L)$, sachant les deux séquences d'observations Y et Y'. Cette solution équivaut à estimer un produit des deux processus (I_l, N_l) et (I'_l, N'_l), avec $A_l = (I_l, I'_l)$. l'algorithme suit le même déroulement que dans la section 4.6. La meilleure séquence (A, N, N') est obtenue à partir des probabilités locales sur les triplets (A_l, N_l, N'_l) grâce à l'équation

$$\mathbb{P}(A, N, N'|Y, Y') = \prod_{l=1}^{L} \mathbb{P}(A_l, N_l, N'_l|Y, Y'), \tag{5.6}$$

qui correspond à l'équation 4.10 de la section 4.6. Le calcul de $\mathbb{P}(A_l, N_l, N'_l|Y, Y')$ est alors effectué par l'intermédiaire de la décomposition

$$\mathbb{P}(A_l, N_l, N'_l|Y, Y') \propto \mathbb{P}(A_l, N_l, N'_l|Y_1^{N_l}, {Y'}_1^{N'_l}) \cdot \mathbb{P}(Y_{N_l+1}^{N}, {Y'}_{N'_l+1}^{N'}|A_l, N_l, N'_l), \tag{5.7}$$

correspondant à l'équation 4.11. Les compteurs N_l et N'_l sont utilisés comme dans la section 3.5.1 pour associer les observations aux symboles. Un calcul en deux passes de type BCJR permet d'obtenir les deux termes de cette équation. La passe avant permet de calculer le premier terme

$$\begin{aligned}\mathbb{P}(A_l = a_l, N_l = n_l, N'_l = n'_l \;\mid\; Y_1^{n_l}, {Y'}_1^{n'_l}) = \sum_{(a_{l-1}, n_{l-1}, {n'}_{l-1})} & \\ \mathbb{P}(A_{l-1} = a_{l-1}, N_{l-1} = n_{l-1}, N'_{l-1} = {n'}_{l-1} \mid Y_1^{n_{l-1}}, {Y'}_1^{{n'}_{l-1}}) \cdot & \\ \mathbb{P}(A_l = a_l \mid A_{l-1} = a_{l-1}) \cdot \mathbb{P}(U_{n_{l-1}+1}^{n_l} = u_{n_{l-1}+1}^{n_l} \mid Y_{n_{l-1}+1}^{n_l}) \cdot & \\ \mathbb{P}({U'}_{{n'}_{l-1}+1}^{{n'}_l} = {u'}_{{n'}_{l-1}+1}^{{n'}_l} \mid {Y'}_{{n'}_{l-1}+1}^{{n'}_l}). & \end{aligned} \tag{5.8}$$

Les termes de cette équation sont respectivement le terme récursif, la probabilité de transition, et la probabilité d'avoir transmis les bits $u_{n_{l-1}+1}^{n_l}$ et ${u'}_{{n'}_{l-1}+1}^{{n'}_l}$ sachant les observations. Ces deux séquences de bits sont entièrement déterminées par la connaissance de a_{l-1} et a_l. La passe arrière permet d'obtenir le second terme de l'équation 5.7

$$\begin{aligned}\mathbb{P}(Y_{n_l+1}^{N}, {Y'}_{{n'}_l+1}^{N'} \;\mid\; A_l = a_l, N_l = n_l, N'_l = n'_l) = \sum_{(a_{l+1}, n_{l+1}, {n'}_{l+1})} & \\ \mathbb{P}(Y_{n_{l+1}+1}^{N}, {Y'}_{{n'}_{l+1}+1}^{N'} \mid A_{l+1} = a_{l+1}, N_{l+1} = n_{l+1}, {N'}_{l+1} = {n'}_{l+1}) \cdot & \\ \mathbb{P}(A_{l+1} = a_{l+1} \mid A_l = a_l) \cdot \mathbb{P}(U_{n_l+1}^{n_{l+1}} = u_{n_l+1}^{n_{l+1}} \mid Y_{n_l+1}^{n_{l+1}}) \cdot & \\ \mathbb{P}({U'}_{{n'}_l+1}^{{n'}_{l+1}} = {u'}_{{n'}_l+1}^{{n'}_{l+1}} \mid {Y'}_{{n'}_l+1}^{{n'}_{l+1}}). & \end{aligned} \tag{5.9}$$

On retrouve les mêmes termes que dans l'équation 5.8.

Nous obtenons finalement un algorithme d'estimation qui a l'avantage d'exploiter toutes les dépendances de la chaîne de codage, mais qui est limité par l'utilisation d'un codeur quasi-arithmétique de précision minimale. Un problème de complexité peut également survenir, du fait de la présence de deux compteurs de bits dans les états estimés. Il est expliqué en section 4.6 que la taille du treillis dépend de l'excursion du compteur. De même, pour le décodage souple de codes de Huffman, les tailles des treillis à horloge bit et à horloge symbole sont conditionnées respectivement par l'excursion du compteur de symboles et du compteur de bits. Nous avons observé expérimentalement dans la première partie de ce chapitre que les tailles des treillis pouvaient devenir excessives. Il en va nécessairement de même pour la technique qui vient d'être présentée.

5.4.2.2 Estimation simplifiée

Un premier moyen pour éviter tout problème de complexité consiste à revenir au décodage séquentiel étudié dans le chapitre 3. Toutefois, nous préférons essayer de concevoir un algorithme d'estimation simplifié basé sur les codes quasi-arithmétiques. L'idée consiste à effectuer l'estimation séparément sur chacune des deux descriptions, puis à fusionner le résultat de ces deux estimations. L'estimation séparée des deux descriptions est effectuée avec l'algorithme de décodage souple étudié dans le chapitre 4. Cet algorithme fournit d'abord en résultat les probabilités a posteriori $\mathbb{P}(X_n, K_n|Y)$ sur les états (X_n, K_n) du modèle produit source + décodeur, puis la séquence d'états (X, K) la plus vraisemblable, qui correspond à la séquence de symboles S décodée.

Pour pouvoir être fusionnée, l'information souple $\mathbb{P}(X_n, K_n|Y)$ et $\mathbb{P}(X'_{n'}, K'_{n'}|Y')$ sur chacune des deux descriptions doit être synchronisée. L'idéal serait de pouvoir convertir cette information sur les états du modèle de décodeur en information sur les symboles. Là encore, nous sommes confrontés à une conversion d'horloge. Pour éviter cette conversion, Nous prenons le parti d'une solution sous-optimale. L'algorithme de décodage souple de codes quasi-arithmétiques nous permet d'obtenir deux séquences de symboles solutions S et S' indépendamment pour chacune des descriptions. Après une conversion de source binaire vers M-aire, nous les fusionnons directement en utilisant la table d'index MDUSQ, qui sert alors de détecteur d'erreurs (section 2.5). Un schéma bloc de ce décodeur est donné par la figure 5.8. Un tel décodeur est symétrique au codeur (figure 5.6).

5.4.2.3 Estimation pseudo itérative

La technique d'estimation simplifiée présentée dans la section précédente a l'avantage d'offrir une complexité réduite. Cependant, lors de l'étape de fusion, les dépendances entre les deux descriptions, ainsi que la corrélation de la source originale ne sont pas complètement exploitées. Dans cette section, nous proposons une piste d'étude pour la conception d'un algorithme d'estimation capable d'exploiter pleinement ces dépendances sans augmenter excessivement la complexité. Il s'agit de s'inspirer du concept de décodage turbo itératif. Au lieu d'itérer entre deux décodeurs de canal, on souhaiterait itérer entre le décodage des deux descriptions. Une application de cette

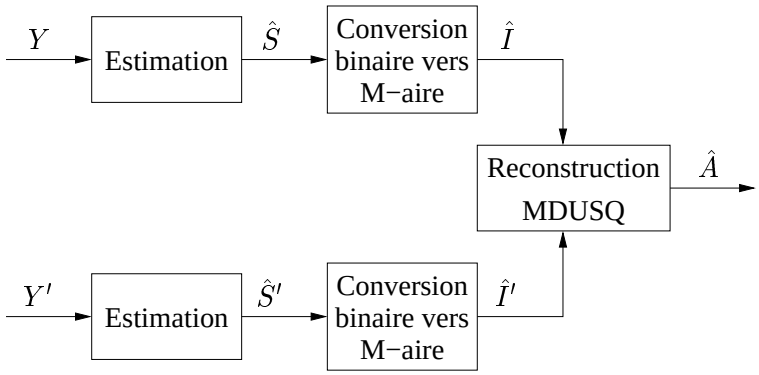


FIG. 5.8 – *Estimation simplifiée de codes quasi-arithmétiques à deux descriptions.*

idée a déjà été proposée par Srinivasan [Sri99], sans la présence du codage entropique. Le problème consiste alors à déterminer l'information souple qui doit être transmise d'un décodeur à l'autre.

Nous considérons dans un premier temps un schéma en deux étapes. Tout d'abord, l'estimation est effectuée séparément sur chacune des deux descriptions, comme dans la section précédente. Les lois a posteriori $\mathbb{P}(X_n, K_n|Y)$ et $\mathbb{P}(X'_{n'}, K'_{n'}|Y')$ sont issues de ces deux estimations. L'étape suivante consiste à échanger cette information souple entre les deux descriptions pour ensuite recommencer deux estimations indépendantes qui permettront d'obtenir les lois a posteriori $\mathbb{P}(X_n, K_n|Y, Y')$ et $\mathbb{P}(X'_{n'}, K'_{n'}|Y, Y')$. Si l'on ignore la présence de codes à longueurs variables, on a après la première étape les lois a posteriori $\mathbb{P}(I_l|Y)$ et $\mathbb{P}(I'_l|Y')$. Le résultat de l'étape suivante peut alors être obtenu par la décomposition

$$\mathbb{P}(I_l|Y, Y') = \mathbb{P}(I_l|Y) \cdot \mathbb{P}(I_l|I'_l) \cdot \mathbb{P}(I'_l|Y'). \tag{5.10}$$

Lorsqu'on utilise le codage quasi-arithmétique, une telle stratégie est encore une fois limitée par la désynchronisation entre les deux descriptions et la nécessité d'effectuer des conversions d'horloge. Si l'on reprend le cas du codeur quasi-arithmétique de précision minimale ($T = 4$), comme en section 5.4.2.1, une telle stratégie devient accessible. En reprenant un modèle à horloge symbole, les lois a posteriori $\mathbb{P}(I_l, N_l|Y)$ et $\mathbb{P}(I'_l, N'_l|Y')$ peuvent être estimée lors de la première étape. L'équation 5.10 augmentée des compteurs N_l et N'_l permet alors d'effectuer la seconde étape. Nous ne sommes cependant toujours pas dans le cadre d'un schéma itératif. Cette idée reste à étudier et à expérimenter.

5.4.3 Résultats expérimentaux

Les performances de l'algorithme de décodage souple de codes MDUSQ+QAC simplifié de la section 5.4.2.2 ont été évaluées expérimentalement sur une source de Gauss-Markov d'ordre 1 de moyenne nulle et de variance unité avec différents facteurs de corrélation ρ. Le débit est de 3 bits par échantillon par description et la largeur de la

diagonale de la table d'index MDUSQ est $d = 2$. La source est donc quantifiée sur 34 niveaux uniformes, ce qui est proche de 5 bits par symbole. Une comparaison avec le décodage souple de codes quasi-arithmétiques est effectuée. Ceux-ci sont appliqués sur une source de Gauss-Markov d'ordre 1 de moyenne nulle et de variance unité quantifiée sur 5 bits. Des marqueurs de synchronisation sont ajoutés de manière à ajuster les débits des deux méthodes. Nous considérons des séquences de longueur $K = 50$ symboles. Toutes les simulations ont été effectuées pour un canal gaussien à bruit blanc additif (AWGN) avec une modulation BPSK. Les résultats sont moyennés sur au moins 400 réalisations.

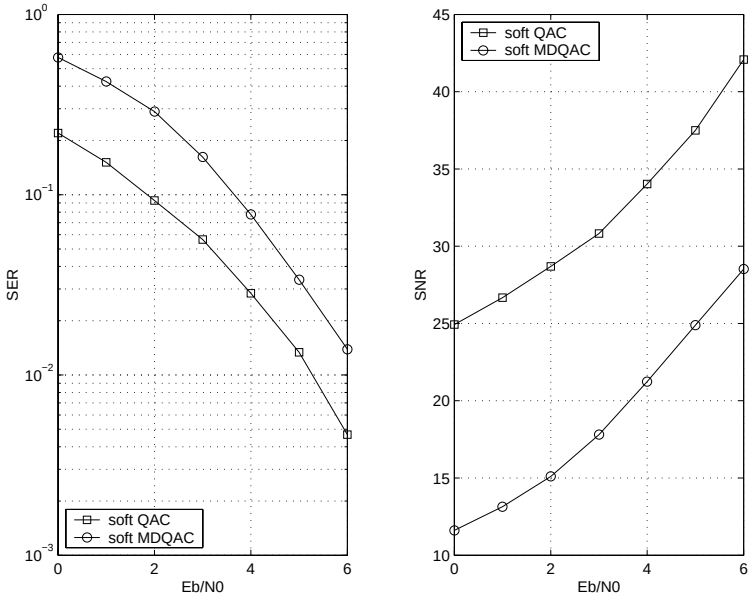


FIG. 5.9 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques ($\rho = 0.1$).*

Les figures 5.9, 5.10 et 5.11 présentent les courbes de SER et SNR résiduels en fonction du rapport signal à bruit E_b/N_0 du canal, respectivement pour $\rho = 0.1$, $\rho = 0.5$ et $\rho = 0.9$. On remarque immédiatement la très nette supériorité du décodage souple de codes quasi-arithmétiques. Cela s'explique par le fait que nous utilisons une technique d'estimation simplifiée pour les codes MDUSQ+QAC. La redondance entre les deux descriptions ainsi que la corrélation de la source ne sont pas complètement exploitées.

Cependant, il ne faut pas oublier que le schéma de codage MDUSQ+QAC est conçu pour résister à la fois aux erreurs et aux effacements. Afin d'évaluer son efficacité dans un tel contexte, nous avons structuré nos données en paquets et introduit un taux de perte de ces paquets dans nos simulations. Chaque paquet contient une séquence complète de symboles codés quasi-arithmétiquement. Dans le cas du codage à simple description, si un paquet est perdu, aucune information concernant son contenu ne subsiste. Dans le cas du codage à descriptions multiples, l'information concernant une séquence de symboles est répartie entre deux paquets, ce qui réduit la probabilité de

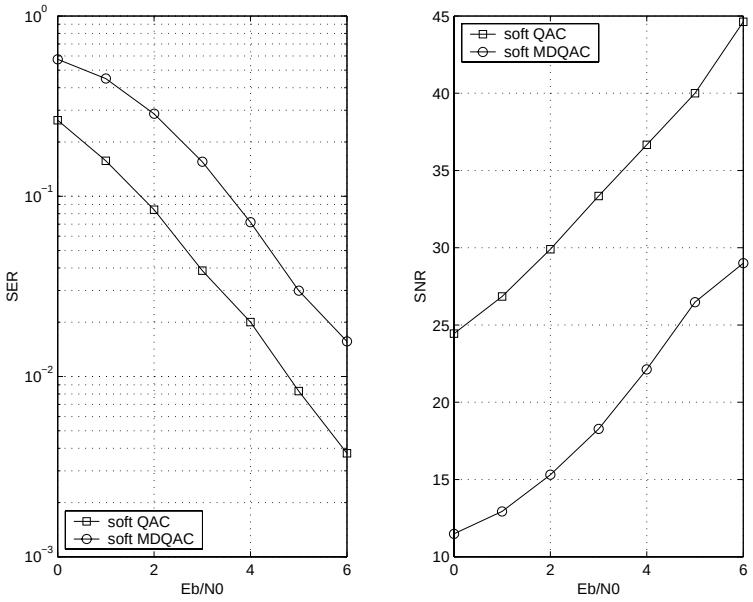


FIG. 5.10 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques* ($\rho = 0.5$).

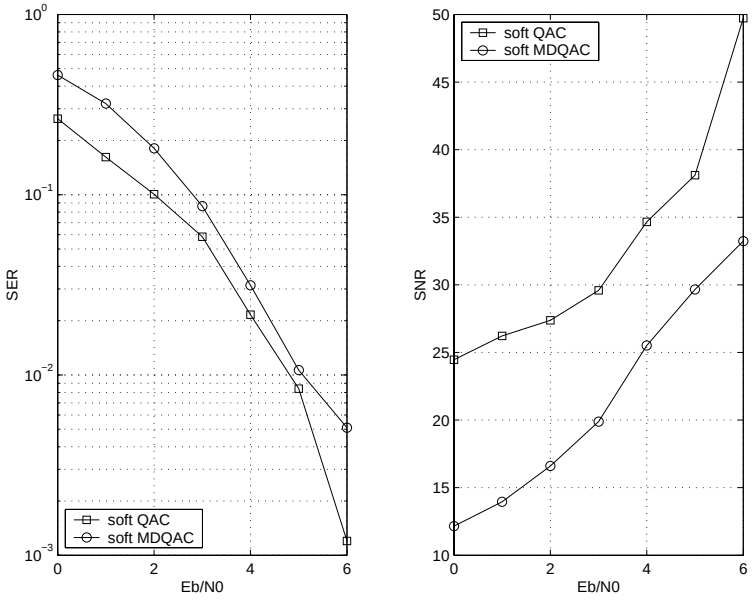


FIG. 5.11 – *SER et SNR pour le décodage souple de codes quasi-arithmétiques et pour le décodage souple de codes MDUSQ + quasi-arithmétiques* ($\rho = 0.9$).

perdre toute information concernant cette séquence. La figure 5.12 illustre les résultats de cette simulation. On constate un bon comportement du codage à descriptions multiples face aux effacements. En revanche, le codage à simple description reste supérieur pour les forts taux d'erreur. Ces résultats peuvent être comparés à ceux de [CKS01], où les auteurs montrent que pour des canaux à erreurs bits, le codage par descriptions multiples est surpassé par des techniques plus classiques où le codage de source et le codage de canal sont optimisés conjointement. Nos résultats confirment ce point, mais montrent également l'intérêt du codage par descriptions multiples lorsque le canal est sujet aux effacements.

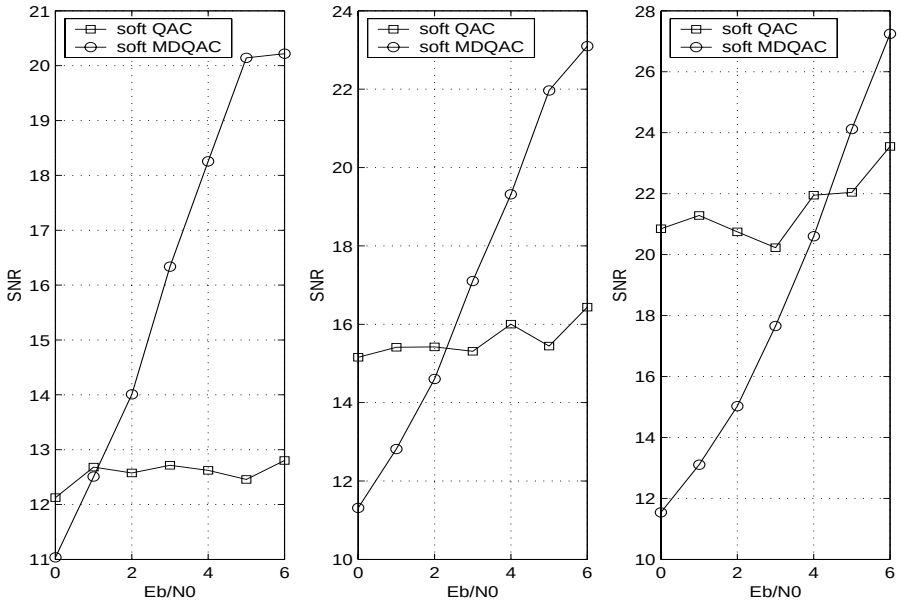


FIG. 5.12 – *SNR pour le décodage souple de codes quasi-arithmétiques avec et sans MDUSQ, pour des taux d'effacement respectivement de 10%, 5% et 1% ($\rho = 0.1$).*

La figure 5.13 combine les résultats de la figure 5.9 (sans effacements) avec ceux de la figure 5.12 (avec effacement). Cela permet d'illustrer la fragilité du codage à simple description et la robustesse du codage à descriptions multiples face aux effacements. En effet, les effacements ne dégradent que légèrement les performances de ce dernier, alors que les pertes sont dramatiques pour le codage à simple description.

5.5 conclusion

Dans le contexte de la transmission sur des canaux sujets à la fois à des erreurs et à des effacements, nous avons proposé la conception de schémas de codage robustes basés sur la combinaison d'une quantification scalaire uniforme à descriptions multiples (MDUSQ) avec des techniques de décodage souple. Le codage par descriptions multiples apporte la robustesse face aux effacements et le décodage souple permet d'éviter la pro-

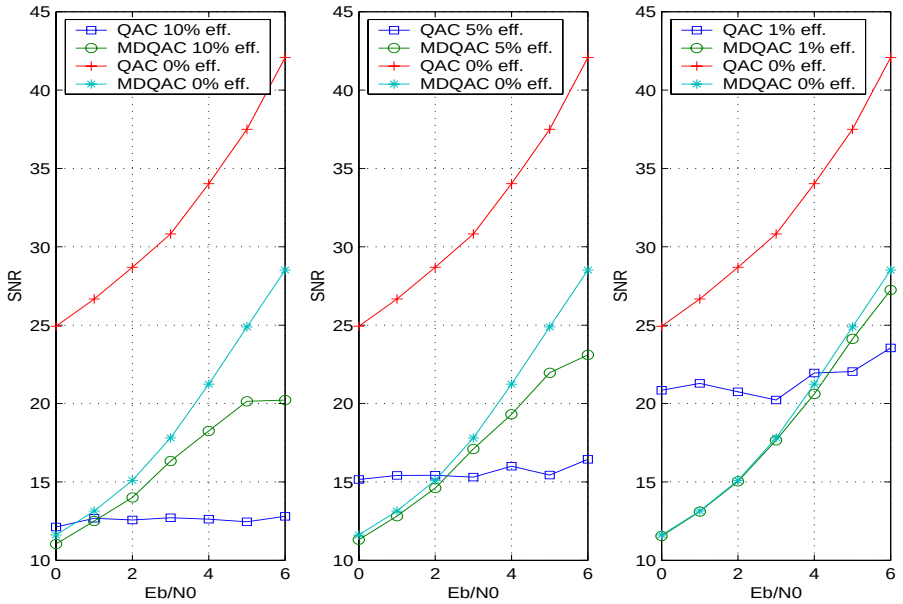


FIG. 5.13 – *SNR pour le décodage souple de codes quasi-arithmétiques avec et sans MDUSQ, avec et sans effacements* ($\rho = 0.1$).

pagation des erreurs dans les codes à longueur variable en favorisant la synchronisation de ces derniers. Le formalisme des réseaux bayésiens permet de modéliser l'ensemble de la chaîne de codage et de concevoir des algorithmes d'estimation qui exploitent la redondance introduite par le codage par descriptions multiples, en plus de la redondance éventuellement ajoutée pour lutter spécifiquement contre les désynchronisations. Le premier schéma proposé est basé sur l'utilisation du codage de Huffman. L'algorithme d'estimation exploite à la fois la corrélation de la source et la redondance entre les deux descriptions. Ensuite, l'utilisation du codage quasi-arithmétique est considérée. La conception de l'algorithme d'estimation s'avère plus délicate dans ce cas, du fait de la difficulté de conversion entre horloge bit et horloge symbole inhérente au codage quasi-arithmétique et de l'absence de synchronisation entre les deux descriptions. Plusieurs algorithmes d'estimation sont alors proposés. Celui qui est retenu offre un bon compromis entre la robustesse et la complexité. Il consiste tout d'abord à effectuer l'estimation indépendamment sur chacune des descriptions, en utilisant l'algorithme du chapitre 4, puis à fusionner le résultat des deux estimations en utilisant la table d'index MDUSQ comme détecteur d'erreur.

Cette dernière approche a été validée expérimentalement dans un contexte de transmission avec effacements et erreurs. Si les résultats obtenus montrent son intérêt, ils montrent également que son efficacité dépend des caractéristiques du canal. Il apparaît, par exemple, que pour de faibles taux d'effacement, le codage à deux descriptions peut être surpassé par le codage à simple description. Les caractéristiques des canaux de transmission ne sont généralement pas stationnaires. Les schémas de codage robustes

ont alors tout intérêt à pouvoir s'adapter à l'état du canal au moment de la transmission, ce qui permet une protection optimale des données. Pour conclure ce chapitre, nous considérons la possibilité de rendre le schéma de codage robuste basé sur la MDUSQ et le décodage souple de codes quasi-arithmétiques adaptable aux conditions de transmission.

Nous faisons l'hypothèse que la source est faiblement corrélée, comme c'est le cas par exemple pour une sous-bande de transformée en ondelettes. Cela implique que s'il est nécessaire de rajouter de la redondance, des marqueurs de synchronisation seront utilisés plutôt qu'un code convolutif. Il se peut que le protocole réseau utilisé inclue un code de canal, que nous n'exploiterons pas. Nous faisons l'hypothèse, en revanche que le codeur et le décodeur disposent d'une information minimale sur l'état du canal, à savoir au moins un taux d'erreur et un taux d'effacement.

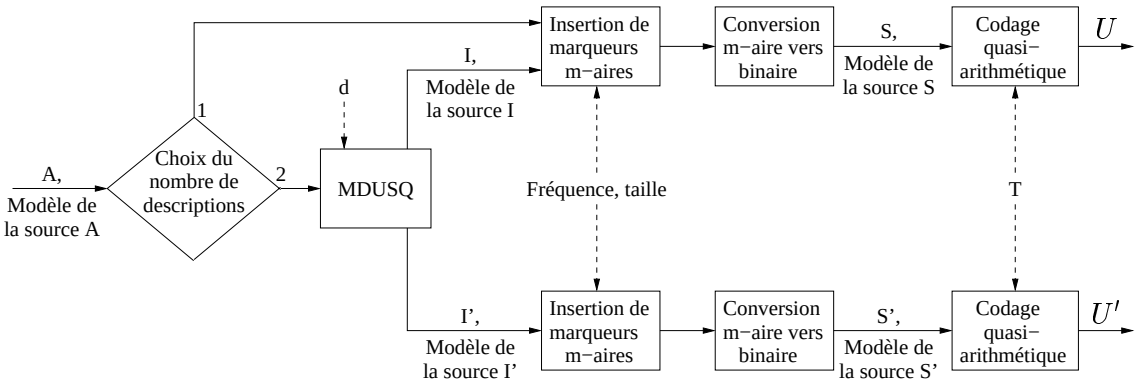


FIG. 5.14 – *Proposition de codeur robuste basé MDUSQ et codage quasi-arithmétique.*

Le fonctionnement du codeur est illustré par le schéma de la figure 5.14. Quatres paramètres peuvent être distingués. Tout d'abord, le choix du nombre de descriptions dépend du taux d'effacement observé sur le canal. Si le codage à deux descriptions est adopté, alors la corrélation entre ces deux descriptions est réglée par le paramètre d, qui représente le nombre de diagonales couvertes dans la table d'index MDUSQ. Ensuite, la fréquence et la taille des marqueurs est réglée en fonction du taux d'erreur observé sur le canal. Notons que sur la figure, les marqueurs signalés sont M-aires, car ils sont insérés avant la conversion de source. Ils peuvent aussi bien être insérés après la conversion. Ce sont alors des marqueurs binaires. Nous préferons les marqueurs M-aires pour une question pratique liée à la mise en oeuvre du calcul du modèle de source binaire. Enfin la précision T du codeur quasi-arithmétique est elle aussi réglée en fonction du taux d'erreur observé.

Au décodage, tous ces paramètres sont supposés connus, ce qui signifie qu'ils ont étés dupliqués dans les deux descriptions et qu'ils ont été protégés par un code de canal approprié. Le fonctionnement du décodeur est illustré sur la figure 5.15. Trois modes de décodage sont possibles. Le décodage séquentiel, basé sur l'algorithme du chapitre 3, le décodage optimal basé sur l'algorithme du chapitre 4 et le décodage instantané.

Conclusion

L'étude menée dans cette thèse s'inscrit dans le contexte général de la transmission de données multimédia sur des réseaux de paquets hétérogènes aux caractéristiques variant dans le temps. En prenant comme point de départ l'étude des techniques de codage par descriptions multiples, nous nous sommes attachés à définir des schémas de codage robustes à la fois aux erreurs et aux effacements et capables de s'adapter aux caractéristiques du canal et des émetteurs/récepteurs. Dans ce chapitre, nous synthétisons l'ensemble des contributions apportées dans ce travail, puis nous donnons quelques perspectives.

5.6 Synthèse

Codage progressif à deux descriptions. Deux codeurs d'image fixe progressifs à deux descriptions basés JPEG2000 ont été développés, l'un basé sur la quantification scalaire uniforme à descriptions multiples (MDUSQ), l'autre basé sur la transformée polyphase associée à la quantification sélective. La caractéristique de progressivité a l'avantage de permettre une adaptation aisée du débit de transmission. Les deux descriptions peuvent contribuer inégalement à la reconstruction de l'image et procurent la robustesse aux effacements. En plus de la progressivité, ces deux codeurs héritent de toutes les caractéristiques de JPEG2000. Dans le cas de la transformée polyphase, la progressivité est utilisée pour l'application du principe de la quantification sélective. Le cas de la MDUSQ a nécessité une étude plus approfondie. En particulier, il a été montré que les tables d'index développées précédemment pour la MDSQ ne sont pas adaptées au codage progressif. Un algorithme de création de table d'index emboîtées a été proposé et à permis une amélioration significative des performances du système. L'impact de la MDUSQ sur les performances du codeur a été étudié.

Décodage souple de codes arithmétiques. Le codage par descriptions multiples ne permet pas en lui même de lutter contre les erreurs. Ce problème a donc été étudié spécifiquement. Le codage entropique, et plus particulièrement le codage arithmétique, a été identifié comme un point faible dans tout système de compression. En effet, une seule erreur bit suffit à désynchroniser un décodeur arithmétique. Ce dernier n'a pas la capacité de se resynchroniser. Tous les symboles qui suivent l'erreur sont alors erronés. Une modélisation bayésienne de la chaîne de codage a été utilisée pour concevoir un

algorithme de décodage souple de codes arithmétiques. La fiabilité de l'estimation a été renforcée grâce à l'ajout et au traitement spécifique de redondance sous forme de marqueurs de synchronisation ou d'information adjacente. L'algorithme de décodage souple de codes arithmétiques, du fait de la structure même de ces codes, souffre d'une complexité exponentielle. Une stratégie d'élagage adaptée permet de contrôler cette complexité, mais rend l'estimation sous-optimale. Finalement, cet algorithme a été intégré dans un codeur JPEG2000. Les résultats expérimentaux montrent des gains de PSNR qui peuvent dépasser les 15 dB.

Décodage souple de codes quasi-arithmétiques. Afin d'éviter l'élagage, des codes arithmétiques simplifiés appelés codes quasi-arithmétiques ont été utilisés. Une modélisation bayésienne de la chaîne de codage a été de nouveau proposée. Les propriétés du codage quasi-arithmétique associées à une modélisation appropriée de la source permettent de représenter le codeur et le décodeur sous la forme d'automates à nombre d'états finis. Un nouvel algorithme d'estimation, optimal cette fois, a alors pu être développé. Les marqueurs de synchronisation et l'information adjacente peuvent également être exploités par cet algorithme. L'utilisation de codes convolutifs a été considérée. Un schéma de décodage itératif inspiré des turbo codes en série a été proposé. Toutes ces approches ont été validées expérimentalement sur des sources de Gauss-Markov. Les résultats dépendent de la corrélation de la source et des taux d'erreur sur le canal. Le décodage souple de codes quasi-arithmétiques surpasse le décodage souple de codes arithmétiques dans la plupart des cas.

Décodage souple de descriptions multiples. Finalement, des schémas de codage robustes à la fois aux effacements et aux erreurs ont été proposés en combinant le codage par descriptions multiples avec le principe du décodage souple. Un premier schéma, basé sur la MDUSQ et les codes de Huffman, a été développé. Dans ce schéma, seule la MDUSQ est utilisée pour ajouter de la redondance. Cette redondance est exploitée par l'estimation. Un deuxième schéma, basé sur la MDUSQ et les codes quasi-arithmétiques a été proposé. Les difficultés de conception d'un algorithme de décodage souple optimal dans ce cas ont été soulignées. Trois techniques d'estimation ont alors été proposées. L'une d'elle a été validée expérimentalement dans un contexte de transmission comportant effacements et erreurs. En conclusion, un schéma général de codage robuste aux effacements et aux erreurs, adaptable aux caractéristiques du canal et des émetteurs/récepteurs a été proposé et discuté.

Résultats. A partir des différents résultats expérimentaux obtenus dans cette thèse, les remarques suivantes peuvent être faites.

- Tout d'abord, les résultats obtenus confirment l'intérêt du codage conjoint source/canal. En effet, nous montrons des situations où l'ajout de redondance directement dans la source est plus efficace que l'ajout de redondance sous forme de codes convolutifs.

- Le type de redondance à ajouter dépend en fait de la corrélation de la source. En effet, pour des sources fortement corrélées, il est inutile de rajouter davantage de corrélation. Des codes correcteurs d'erreurs sont alors plus efficaces. En revanche, si la source est faiblement corrélée, il est préférable d'augmenter cette corrélation plutôt que d'utiliser des codes correcteurs d'erreurs.
- Enfin pour finir, nous remarquons qu'il est toujours intéressant de compresser davantage les données. Dans le cadre du développement des techniques de codage conjoint source/canal, une attitude possible consiste à dire qu'il est préférable de compresser moins les données afin de garantir une plus grande robustesse. Par exemple, les codes de Huffman sont moins efficaces que les codes arithmétiques, mais permettent le décodage souple optimal. Or nous montrons des résultats où le décodage souple non optimal de codes arithmétiques surpasse celui optimal des codes de Huffman. Cela s'explique par le fait que le gain en compression des codes arithmétiques a été exploité pour ajouter une forme de redondance qui favorise la synchronisation du code. Il est donc dans ce cas plus intéressant de compresser plus et d'exploiter le gain de compression pour le traitement approprié des erreurs. Cette remarque doit tout de même être relativisée, puisque nous montrons aussi des cas où elle n'est pas vérifiée.

5.7 perspectives

A la suite des travaux effectués dans cette thèse, de nombreux développements sont possibles, parmi lesquels :

- **Le décodage souple de codes quasi-arithmétiques adaptatifs**. Le codage quasi-arithmétique adaptatif a été mentionné, mais n'a pas été traité. Ce thème n'est pas à négliger, si l'on considère le fait que des standards de compression récents tels que H264 incluent le codage arithmétique adaptatif.
- **L'utilisation d'autres techniques de codage par descriptions multiples**. Dans cette thèse, la MDUSQ a été adoptée pour ses bonnes performances, son réglage aisé de la redondance et parce qu'elle permet de modéliser facilement les dépendances entre les descriptions. Cependant, suivant l'application, d'autres techniques peuvent être associées au principe du décodage souple, telles que les techniques spectrales, la transformée polyphase, les bancs de filtres redondants et d'autres encore.
- **L'introduction des effacements dans le décodage souple**. Dans nos simulations, nous avons considérés que les effacements étaient corrigés uniquement par le codage par descriptions multiples. C'est à dire que les descriptions reçues contiennent forcément les observations d'une séquence de bits codée entière. Les algorithmes d'estimations utilisés peuvent être adaptés au cas d'effacements "locaux". Pour une séquence de bits donnée, si un sous ensemble de ces bits n'est pas reçu, l'estimation peut tout de même être effectuée en mettant à 0.5 les probabilités des bits non observés. La fiabilité de l'estimation dans ce cas reste à étudier. Des expériences préliminaires ont été réalisées et des résultats prometteurs ont

été obtenus.

- **le décodage souple rapide**. Dans les techniques de décodage souple considérées, la complexité peut rapidement devenir critique, du fait en particulier de la taille des treillis d'estimation. La complexité de l'algorithme de décodage souple de codes arithmétiques est contrôlée grâce à l'élagage. Des mises en oeuvre rapide sont possibles, et pourrait être développées. Cet algorithme fonctionne aussi bien avec les codes quasi-arithmétiques. Il serait intéressant, cependant, de redéfinir un algorithme d'estimation et une stratégie d'élagage spécifiquement pour ces derniers.
- **L'application au codage vidéo**. La conception d'un codeur vidéo "scalable" et robuste à la fois aux effacements et aux erreurs est encore un défi. La piste du codage vidéo en sous-bandes 3D semble particulièrement intéressante. Des bancs de filtres redondants peuvent être utilisés pour la création de ces sous-bandes. Le décodage souple n'est envisageable qu'avec une mise en oeuvre rapide.
- **L'évaluation théorique des performances du décodage souple**. Nous avons souligné la difficulté d'évaluer théoriquement l'impact du décodage souple sur le taux d'erreur symbole ou sur la distorsion en fonction d'un taux d'erreur bit. La résolution de cette question permettrait de concevoir des fonctions d'optimisation d'allocation de la redondance pour le décodage souple.
- **L'adoption du codage quasi-arithmétique dans un standard de compression**. Il pourrait être particulièrement intéressant d'adopter le codage quasi-arithmétique à la place du codage arithmétique dans un standard de compression d'image ou de vidéo. En effet, dans la mesure où la précision du code est paramétrable cela n'aurait aucun coût, excepté quelques bits d'en-tête global pour transmettre le paramètre de précision. Seulement quatre valeurs de la précision peuvent suffire ($T = 4, 8, 16, max$). Dans le cas de la précision maximale, les performances en compression sont conservées par rapport au codage arithmétique, sans inconvénient particulier. Si la précision est réduite, cela ouvre la possibilité, d'une part d'offrir une robustesse accrue grâce au décodage souple, et d'autre part de permettre des mises en oeuvre rapide grâce à la modélisation sous forme d'automate à nombre d'états fini. Dans ce dernier cas, une simple table (LUT) suffit à effectuer le codage et le décodage. Notons que dans le cas où la robustesse est primordiale, des marqueurs de synchronisation peuvent être exploités. De tels marqueurs sont déjà intégrés dans le standard JPEG2000.

Annexe A

JPEG 2000

A.1 Introduction

L'utilisation des images numériques est de plus en plus importante dans de nombreux domaines et la qualité de ces images a considérablement augmentée. Plus que jamais, il est nécessaire de compresser ces images pour les stocker et les transmettre. Mais il est aussi nécessaire de disposer de fonctionnalités nouvelles pour éditer, traiter et transmettre ces images sur des réseaux et des terminaux hétérogènes. Ce constat a motivé la création du standard JPEG2000, qui a été proposé pour prendre la suite du standard JPEG. Ce standard a été finalisé en décembre 2000 [ISO00].

Le standard JPEG2000 n'a pas pour seul intérêt des performances en compression améliorées. Il propose également un certain nombre de fonctionnalités telles que :

- la transmission progressive par qualité, résolution ou composante.
- La compression avec ou sans perte.
- L'accès aléatoire au train binaire et la possibilité de décodage d'une partie seulement de l'image.
- Le traitement dans le domaine compressé (par exemple, rotation ou rognage).
- Le codage prioritaire de régions d'intérêts.

L'historique du développement du standard JPEG2000 est résumé dans [MGBB00]. Des détails supplémentaires sont également fournis sur son fonctionnement et ses fonctionnalités.

La structure de base d'un codeur JPEG2000 est résumée par la figure 2.1, en section 2.1. Ce codeur est composé d'une transformée en ondelettes, d'une quantification et d'un algorithme spécifique de codage emboîté appelé *EBCOT (Embedded Block Coding with Optimized Truncation)*. Ce dernier, proposé par Taubman [Tau00], constitue l'élément clé de JPEG2000. EBCOT est une évolution de l'algorithme LZC (*layered zero coding*) [TZ94]. Notons que la version d'EBCOT utilisée dans le standard JPEG2000 est légèrement différente de celle proposée originalement par Taubman. La différence majeure entre EBCOT et d'autres codeurs tels que EZW ou SPIHT réside dans le découpage en blocs des sous-bandes de la transformée en ondelettes. La où EZW et

SPIHT utilisent la corrélation inter sous-bandes pour la compression, EBCOT n'exploite que la corrélation intra bloc. Nous détaillons à présent le fonctionnement d'EBCOT.

A.2 Codage EBCOT

L'algorithme EBCOT est un algorithme de codage emboîté d'une décomposition en ondelettes discrète d'une image. EBCOT peut indifféremment prendre en entrée une représentation multirésolution telle que celle de Mallat, ou bien une décomposition quelconque en paquets d'ondelettes.

Dans EBCOT, chaque sous-bande est partitionnée en blocs d'échantillons B_i, généralement de taille 64×64, mais qui peut varier. EBCOT génère pour chaque bloc un train binaire indépendant. Chaque train binaire ainsi formé est hautement scalable, c'est à dire qu'il peut être tronqué à des points R_i^n, correspondant respectivement à une distorsion D_i^n. La propriété intéressante de cet ensemble de points de troncature (R_i^n, D_i^n) est que la plupart de ces points sont situés sur l'enveloppe convexe de la courbe débit-distorsion du bloc considéré.

Afin de générer un train binaire progressif pour représenter l'image entière, EBCOT organise le train binaire final en couches de qualité $\mathcal{Q}_q$. Étant donné un débit alloué à une couche donnée, le train binaire de chacun des blocs est tronqué de façon à minimiser la distorsion globale associée au décodage de la couche considérée. L'algorithme d'optimisation débit-distorsion, qui détermine la contribution de chaque bloc aux différentes couches, est appelé PCRD (*Post Compression Rate-Distortion*). L'organisation des couches de qualité est illustrée par la figure A.1.

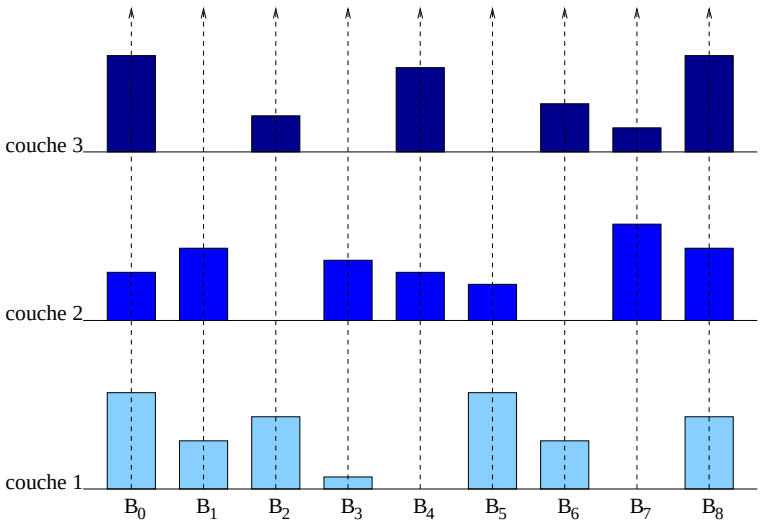


FIG. A.1 – *Organisation des couches de qualité dans EBCOT. Par exemple, le bloc* 7 *ne contribue pas à la couche* 1.

En plus des portions des trains binaires de blocs concaténés, les couches de qualité $\mathcal{Q}_q$ contiennent des informations auxiliaires indiquant le point de troncature n_i associé à chaque bloc contribuant à la couche, ainsi que la longueur de train binaire $R_i^{n_i}$ qui y correspond. Une deuxième brique est introduite dans EBCOT, destinée à compresser ces données auxiliaires et à structurer l'information en *paquets EBCOT*. Ceux-ci sont conçus de façon à pouvoir réordonner le train binaire en fonction du type de progressivité désiré. Ils permettent également l'accès aléatoire aux données. L'ensemble de l'algorithme EBCOT est schématisé par la figure A.2.

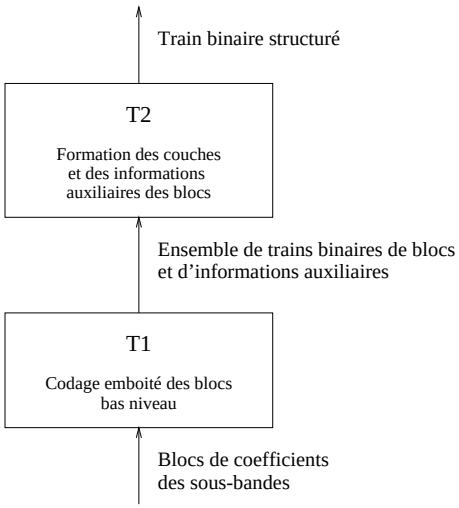


FIG. A.2 – *Briques principales de l'algorithme EBCOT.*

A.2.1 Algorithme PCRD

L'algorithme PCRD tente de trouver, pour chaque bloc B_i les points de troncature n_i de façon à minimiser la distorsion supposée additive:

$$D = \sum_i D_i^{n_i} \tag{A.1}$$

sous la contrainte

$$R = \sum_i R_i^{n_i} \ \leq \ R^{max} \tag{A.2}$$

La mesure de distorsion $D_i^{n_i}$ correspond à l'erreur quadratique moyenne pondérée par le carré de la norme L_2 des fonctions d'ondelettes de base pour la sous-bande b_i, à laquelle appartient le bloc B_i. Une approche lagrangienne est adoptée, consistant à minimiser la grandeur suivante:

$$(D(\lambda) + \lambda R(\lambda)) \ = \ \sum_i \left(D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda} \right) \tag{A.3}$$

Pour ce faire, étant donné l'ensemble de tous les points de troncature possibles pour le bloc B_i : $j_1 < j_2 < j_3 < \ldots$, définissons les pentes distorsion-débit suivantes:

$$S_i^{j_k} = \frac{\Delta D_i^{j_k}}{\Delta R_i^{j_k}} = \frac{D_i^{j_{k-1}} - D_i^{j_k}}{R_i^{j_k} - R_i^{j_{k-1}}} \tag{A.4}$$

Un ensemble de points de troncature $j_1 < j_2 < j_3 < \ldots$ est dit admissible si la suite des pentes correspondantes $S_i^{j_1} < S_i^{j_2} < \ldots$ est strictement croissante. L'ensemble $\mathcal{N}_i$ de points de troncature admissible le plus grand pour un bloc B_i est formé après le codage emboîté de chaque bloc (brique T_1 de EBCOT). Ainsi, minimiser le lagrangien de l'équation A.3 revient simplement à trouver:

$$n_i^\lambda = \max\left\{ j_k \in \mathcal{N}_i | S_i^{j_k} > \lambda \right\} \tag{A.5}$$

A.2.2 Codage de blocs emboîté

Quantification Tout d'abord, la formation pour chaque bloc B_i d'un train binaire emboîté utilise une quantification scalaire avec zone morte (quantification *dead-zone*). Nous adoptons dans la suite les notations suivantes:

- $s_i[\mathbf{k}] = s[k_1, k_2]$: séquence d'échantillons de sous-bandes appartenant au bloc B_i (k_1 et k_2 sont les positions horizontale et verticale)
- $\chi_i[\mathbf{k}] \in \{-1, 1\}$: signe de $s_i[\mathbf{k}]$
- $\nu_i[\mathbf{k}] = \frac{|s_i[\mathbf{k}]|}{\delta_{\beta_i}}$: amplitude des échantillons quantifiés, représentée sur M_i bits.
- δ_β est le pas de quantification de la sous-bande β et β_i est la sous-bande contenant le bloc B_i.
- $\nu_i^p[\mathbf{k}]$: $p^{\text{ème}}$ bit de la représentation binaire de $\nu_i[\mathbf{k}]$.

Signifiance des sous-blocs Ce point n'apparaît pas dans le standard, mais fait partie de l'algorithme EBCOT original. L'idée consiste à découper les blocs en sous blocs, par exemple de taille 16×16 et d'utiliser un codage par *quadtree* pour coder efficacement les sous blocs qui ne contiennent que des 0.

Primitives de codage de plans de bits La signifiance $\sigma_i[\mathbf{k}]$ d'un échantillon $\mathbf{k}$ est définie comme suit: $\sigma_i[\mathbf{k}] = 0$ tant que le premier bit $\nu_i^p[\mathbf{k}]$ non nul de $\nu_i[\mathbf{k}]$ n'a pas été codé. Quatre primitives sont alors utilisées pour coder les plans de bits:

- si $\sigma_i[\mathbf{k}] = 0$: utilisation d'une combinaison des primitives ZC (Zero Coding) et RLC (Run-Length Coding) pour coder si on a $\nu_i^p[\mathbf{k}] = \mathbf{1}$ ou non;
- si $\sigma_i[\mathbf{k}] = \mathbf{0}$ et $\nu_i^p[\mathbf{k}] = \mathbf{1}$: utilisation de la primitive SC (Sign Coding) pour coder le signe de $\nu_i[\mathbf{k}]$;
- si $\sigma_i[\mathbf{k}] = \mathbf{1}$: utilisation de la primitive MR (Magnitude Refinement) pour coder la valeur du nouveau bit $\nu_i^p[\mathbf{k}]$ (où p est le numéro du plan de bits courant);

Contextes associés au codage des primitives Les dépendances statistiques entre signifiances d'échantillons sont capturées via la formation de trois types de contextes différents (processus illustré figure A.3):

- h : nombre de voisins horizontaux signifiants
- v : nombre de voisins verticaux signifiants
- d : nombre de voisins diagonaux signifiants

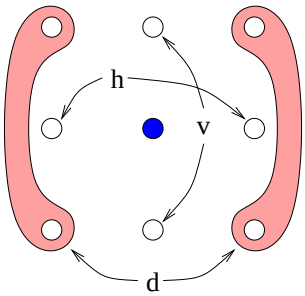


FIG. A.3 – *Contextes formés pour le codage arithmétique des primitives*

A.2.3 Plans de bits partiels

La figure A.4(a) représente les points du plan débit-distorsion obtenus lorsqu'un codage par plans de bits avec le codeur EBCOT est effectué. La courbe d'interpolation montre que tronquer un plan de bit donné avant son décodage complet conduit à des performances débit-distorsion sous-optimales. Une technique de *plans de bits partiels* est utilisée dans EBCOT, afin d'obtenir un train binaire finement granulaire, fournissant une multitude de points de troncature situés sur la courbe débit-distorsion de la source.

L'idée des plans de bits partiels consiste à séparer les échantillons du code-block traité en sous-ensembles statistiquement distincts. Il est alors possible de coder les échantillons par sous-ensembles, en transmettant d'abord les sous-ensembles conduisant à la baisse de distorsion la plus importante. Dans le cas de EBCOT, 3 sous-ensembles sont construits et correspondent à 3 passes de codage différentes, notées $\mathcal{P}_1^p, \ldots, \mathcal{P}_3^p$, où la fin de $\mathcal{P}_3^p$ marque le point auquel tous les échantillons ont été mis à jour dans le plan de bits p. Notons que l'algorithme EBCOT original comporte une passe de plus.

Brièvement, les rôles joués respectivement par chaque passe sont les suivants.

- "Forward Signifiance Pass", $\mathcal{P}_1^p$:
  - Suivi d'un parcours défini comme suit : pour chaque échantillon $s_i[\mathbf{k}]$ insignifiant qui a un voisin signifiant, application de la primitive ZC, accompagnée de la primitive SC si $\nu_i^p[\mathbf{k}] = \mathbf{1}$.
  - Les autres échantillons sont ignorés.
  - Pour chaque échantillon traité, mise à 1 de la variable d'état $\eta_i[\mathbf{k}]$ (initialisée à 0 à chaque passe de normalisation $\mathcal{P}_3^p$).

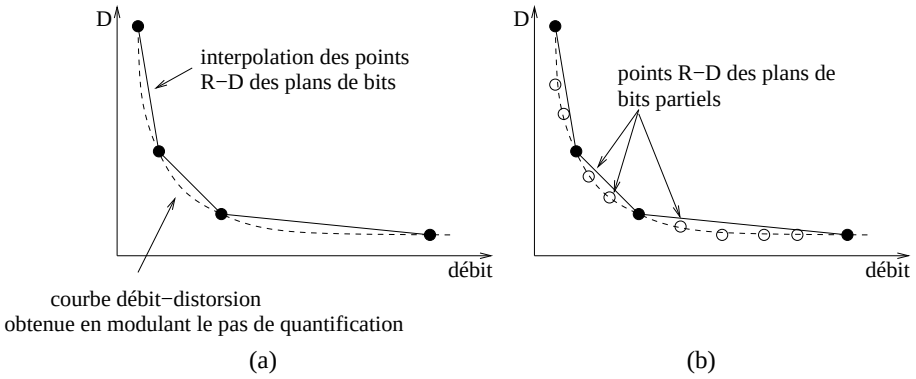


FIG. A.4 – *Propriétés débit-distorsion (a) du codage par plans de bits régulier et (b) du codage par plans de bits partiels.*

- "Magnitude Refinement Pass", $\mathcal{P}_2^p$:
  - Suivi d'un parcours défini comme suit : Pour chaque coefficient signifiant pour lequel aucune information n'a été codée ($\sigma_i[\mathbf{k}] = \mathbf{1}$ et $\eta_i[\mathbf{k}] = \mathbf{0}$) dans le plan de bits p courant, utilisation de la primitive MR.
- "Normalization Pass", $\mathcal{P}_3^p$:
  - Parcours de tous les échantillons qui n'ont pas encore été traités ($\eta_i[\mathbf{k}] = \mathbf{0}$ et $\sigma_i[\mathbf{k}] = \mathbf{0}$).
  - Traitement des échantillons considérés avec les primitives ZC, RLC et SC si nécessaire.

L'ordre d'apparition des différentes passes dans l'organisation du train binaire final est illustré figure A.5. On remarque que le train binaire commence par une passe de normalisation. En effet, les deux autre passes ont besoin d'une initialisation du processus pour fonctionner.

$\mathcal{P}_3^{p_{max_i}}$	$\mathcal{P}_1^{p_{max_i}-1}$	$\mathcal{P}_2^{p_{max_i}-1}$	$\mathcal{P}_3^{p_{max_i}-1}$	...	$\mathcal{P}_1^0$	$\mathcal{P}_2^0$	$\mathcal{P}_3^0$

FIG. A.5 – *Ordre des passes dans le train binaire EBCOT final d'un bloc B_i.*

Finalement, l'algorithme de codage progressif EBCOT présente des performances au moins égales à ses concurrents EZW et SPIHT. EBCOT fournit par ailleurs plus de flexibilité, telles que la scalabilité en résolution (EZW et SPIHT ne fournissent que la scalabilité SNR), ou bien des fonctionnalités d'accès aléatoire aux données dans le domaine compressé.

Annexe B

Réseaux bayésiens et inférence

B.1 introduction

Dans les chapitres 3, 4 et 5, nous analysons la structure de chaînes de codages dans le contexte des réseaux bayésiens, suivant la méthodologie déjà employée dans [GFGR01]. Les réseaux bayésiens constituent un outil naturel pour représenter la structure à la fois des dépendances stochastiques et des contraintes entre des variables aléatoires. Ils permettent également de mettre en évidence les relations d'indépendance conditionnelle dans un modèle. Or, ces relations sont primordiales pour la conception d'algorithmes d'estimation rapides. En effet, la structure d'un algorithme d'inférence bayésienne, soit exact, soit approché, peut se déduire "automatiquement" de la représentation graphique du modèle. De tels algorithmes s'obtiennent en combinant certaines opérations primitives : la *propagation*, la *mise à jour* et la *fusion*. Il existe de nombreux agencements possibles de ces primitives, ce qui rend difficile la classification des stratégies d'estimation, hormis si l'on considère le critère sur lequel elles se basent. De plus, de nombreuses communautés scientifiques ont découvert indépendamment certaines de ces stratégies, d'où une large variété de noms pour des algorithmes très similaires (par exemple l'algorithme BCJR et la propagation de croyance).

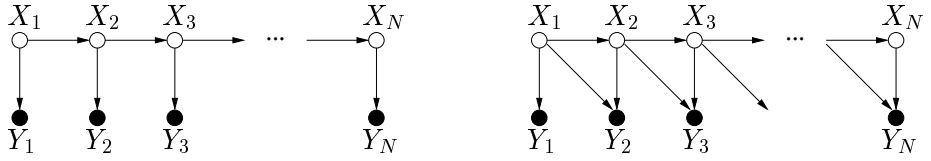


FIG. B.1 – *Processus de Markov X et mesure associée Y (a) sur les variables de X et (b) sur les transitions de X.*

Dans cette annexe, nous nous appuyons sur [GFGR01] pour présenter certaines de ces stratégies d'estimation appliquées à un processus de Markov. On peut citer [GCSR95] et [Lau96] comme lectures complémentaires sur ce sujet. Les algorithmes sont classés d'abord suivant le critère d'estimation, puis suivant l'organisation des cal-

culs. Le processus de Markov à estimer est $X = X_1 \dots X_N$. La factorisation $\mathbb{P}(X) = \prod_n \mathbb{P}(X_n|X_{n-1})$ est représentée graphiquement par la chaîne orientée de la figure B.1. Le processus X est caché. L'information le concernant est obtenue à travers le processus de mesure éventuellement bruité $Y = Y_1 \dots Y_N$. Les mesures sont supposées indépendantes *conditionnellement* à X. On a donc $\mathbb{P}(Y|X) = \prod_n \mathbb{P}(Y_n|X)$. Les mesures sont également supposées *locales*, ce qui signifie que Y_n mesure uniquement X_n ($\mathbb{P}(Y_n|X) = \mathbb{P}(Y_n|X_n)$), ou bien la transition entre X_{n-1} et X_n ($\mathbb{P}(Y_n|X) = \mathbb{P}(Y_n|X_{n-1}, X_n)$).

B.2 Estimation MPM

B.2.1 Stratégies "organisées"

MPM est l'abréviation de *maximum of posterior marginals (maximum des marginales à posteriori)*, ce qui signifie que chaque X_n est estimé individuellement suivant

$$\hat{X}_n = \arg\max_{x_n} \mathbb{P}(X_n = x_n|Y). \tag{B.1}$$

Par conséquent, l'objectif est de calculer les marginales a posteriori $\mathbb{P}(X_n|Y)$. Le calcul peut s'organiser à partir de la factorisation

$$\mathbb{P}(X_n|Y) \propto \mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(Y_{n+1}^N|X_n), \tag{B.2}$$

où $\propto$ désigne un facteur de normalisation et où $Y_a^b = Y_a, Y_{a+1}, \dots, Y_b$. La propriété de Markov permet un calcul récursif de chacun des termes de la partie droite de l'équation. La *passe avant* permet d'obtenir le premier terme

$$\mathbb{P}(X_n|Y_1^n) \propto \mathbb{P}(X_n|Y_1^{n-1}) \cdot \mathbb{P}(Y_n|X_n), \tag{B.3}$$

où

$$\mathbb{P}(X_n|Y_1^{n-1}) = \sum_{x_{n-1}} \mathbb{P}(X_n|X_{n-1} = x_{n-1}) \cdot \mathbb{P}(X_{n-1} = x_{n-1}|Y_1^{n-1}). \tag{B.4}$$

L'équation B.4 est généralement appelée étape de *propagation* ou de *prédiction*. L'équation B.3, quant à elle, est appelée étape de *mise à jour*. Nous avons supposé que Y_n mesurait X_n. Si Y_n mesure la transition entre X_{n-1} et X_n, alors il faut remplacer $\mathbb{P}(Y_n|X_n)$ par $\mathbb{P}(Y_n|X_{n-1}, X_n)$. Nous considérerons dorénavant que nous sommes dans le premier cas. La *passe arrière* permet de calculer le second terme de l'équation B.2

$$\mathbb{P}(Y_{n+1}^N|X_n) \propto \sum_{x_{n+1}} \mathbb{P}(X_{n+1} = x_{n+1}|X_n) \cdot \mathbb{P}(Y_{n+2}^N|X_{n+1} = x_{n+1}) \cdot \mathbb{P}(Y_{n+1}|X_{n+1} = x_{n+1}). \tag{B.5}$$

Comme cette quantité tend vers zéro avec l'augmentation du nombre de mesures, elle est souvent manipulée sous une forme renormalisée sur la variable X_n, d'où le symbole

$\propto$, qui n'a pas d'influence sur l'équation B.2. Cette organisation en deux passes des calculs est celle utilisée dans l'algorithme BCJR. Elle est bien adaptée à la définition de $\mathbb{P}(X)$ à travers la probabilité de transition $\mathbb{P}(X_n|X_{n-1})$. Une version plus symétrique évite le changement d'orientation du second terme de l'équation B.2 :

$$\mathbb{P}(X_n|Y) \propto \frac{\mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(X_n|Y_{n+1}^N)}{\mathbb{P}(X_n)}. \tag{B.6}$$

Cette factorisation constitue une fusion de deux distributions conditionnelles latérales de X_n. Le second terme du numérateur peut être obtenu à partir des équations B.3 et B.4. L'équation B.6 nécessite la connaissance non seulement de $\mathbb{P}(X_n|X_{n-1})$ mais aussi de $\mathbb{P}(X_n)$.

Remarque: les marginales a posteriori sur les transitions $\mathbb{P}(X_n, X_{n+1}|Y)$ sont un sous-produit direct de l'algorithme d'estimation MPM. Par exemple, si la factorisation de l'équation B.2 a été choisie, on a

$$\mathbb{P}(X_n, X_{n+1}|Y) \propto \mathbb{P}(X_n|Y_1^n) \cdot \mathbb{P}(X_{n+1}|X_n) \cdot \mathbb{P}(Y_{n+1}|X_{n+1}) \cdot \mathbb{P}(Y_{n+1}^N|X_{n+1}). \tag{B.7}$$

B.2.2 Stratégies "aveugles"

Les stratégies d'estimation vues précédemment sont organisées en deux passes ou récursions. Elle s'appuient sur le graphe de la figure B.1. D'autres algorithmes, au contraire, spécifient uniquement les calculs locaux et laissent l'organisation de la circulation des messages indéfinie. Puisque aucune connaissance globale du graphe n'est nécessaire, cette famille d'algorithme peut être désignée par *stratégies aveugles*.

L'idée consiste à atteindre $\mathbb{P}(X_n|Y)$ en augmentant progressivement le nombre de mesures dans la partie conditionnelle de $\mathbb{P}(X_n|Y_I)$, avec $I \subset \{1, 2, \ldots, N\}$. Soient les deux ensembles d'indices $L(n) \subset \{1, \ldots, n-1\}$ et $R(n) \subset \{n+1, \ldots, N\}$. comme $Y_{L(n)}$, Y_n et $Y_{R(n)}$ sont indépendants conditionnellement à X_n, on a l'équation de fusion

$$\mathbb{P}(X_n|Y_{L(n)}, Y_n, Y_{R(n)}) \propto \frac{\mathbb{P}(X_n|Y_{L(n)}) \cdot \mathbb{P}(X_n|Y_n) \cdot \mathbb{P}(X_n|Y_{R(n)})}{\mathbb{P}(X_n)^2}. \tag{B.8}$$

Pour chaque X_n, l'algorithme passe des messages $\mathbb{P}(X_n|Y_I)$ sur les arcs qui entourent X_n. Les Y_i sont des mesures situées au delà de ces arcs, du point de vue de X_n. Par exemple, l'arc (X_{n-1}, X_n) apporte l'information $\mathbb{P}(X_n|Y_{L(n)})$ à X_n. Ces messages sont mis à jour par fusion et propagés. Le message $\mathbb{P}(X_{n-1}|Y_{R(n-1)})$ envoyé par X_n à X_{n-1} est mis à jour par

$$R(n-1) = R(n) \cup \{n\}, \tag{B.9}$$

$$\mathbb{P}(X_{n-1}|Y_{R(n-1)}) \propto\propto \sum_{x_n} \mathbb{P}(X_{n-1}|X_n = x_n) \cdot \frac{\mathbb{P}(X_n = x_n|Y_n) \cdot \mathbb{P}(X_n = x_n|Y_{R(n)})}{\mathbb{P}(X_n) = x_n}. \tag{B.10}$$

Le message envoyé à X_{n+1} est obtenu symétriquement. L'équation B.10 peut être généralisée en laissant l'inclusion de Y_n optionnelle. Par monotonie, le seul état stable de l'algorithme est obtenu pour $L(n) = \{1, \ldots, n-1\}$ et $R(n) = \{n+1, \ldots, N\}$ pour tout n. Par conséquent, quel que soit l'ordre des mises à jour, l'équation B.8 permet finalement d'obtenir les marginales a posteriori souhaitées. Notons que des variations de cet algorithme peuvent être obtenues en définissant les messages et les mises à jour à partir d'autres factorisations, comme par exemple

$$\mathbb{P}(X_n|Y_{L(n)}, Y_n, Y_{R(n)}) \propto \mathbb{P}(Y_{L(n)}|X_n) \cdot \mathbb{P}(Y_n|X_n) \cdot \mathbb{P}(Y_{R(n)}|X_n) \cdot \mathbb{P}(X_n). \tag{B.11}$$

Notons également que la stratégie organisée présentée précédemment constitue un cas particulier de stratégie aveugle. Enfin on peut signaler que quand Y mesure les transitions de X, l'index n de la mesure Y_n doit être compté par $L(n)$.

B.3 Estimation MAP

Le critère du MAP, ou *maximum a posteriori* correspond à l'estimation bayésienne optimale de la totalité du processus X sachant les mesures disponibles Y :

$$\hat{X} = \arg\max_x \ \mathbb{P}(X = x|Y). \tag{B.12}$$

Par conséquent, l'optimisation est effectuée pour *toutes les séquences x possibles.* soient $I \subset \{1, \ldots, N\}$ un ensemble d'index et $J = I \backslash \{n\}$. La notation $\bar{\mathbb{P}}$ est définie par

$$\bar{\mathbb{P}}(X_n|Y_I) \propto \max_{x_J} \mathbb{P}(X_n, X_J = x_j|Y_I), \tag{B.13}$$

$$\bar{\mathbb{P}}(Y_I|X_n) \propto \max_{x_J} \mathbb{P}(Y_I, X_J = x_j|X_n). \tag{B.14}$$

Cette définition permet d'éliminer une constante multiplicative qui est inutile pour l'estimation. Cela permet des renormalisations sur X_n qui favorisent la stabilité de l'algorithme. La séquence optimale $\hat{X}$ peut s'obtenir en réunissant les estimées locales $\hat{X}_n$ définies par

$$\hat{X}_n = \arg\max_{x_n} \ \bar{\mathbb{P}}(X_n = x_n|Y). \tag{B.15}$$

Par conséquent, l'objectif est de nouveau d'obtenir ces marginales a posteriori. Les factorisations de Bayes telles que celles des équations B.2, B.6, B.8 ou B.11 restent valides pour $\bar{\mathbb{P}}$. Par exemple, l'équation B.2 devient

$$\bar{\mathbb{P}}(X_n|Y) \propto \bar{\mathbb{P}}(X_n|Y_1^n) \cdot \bar{\mathbb{P}}(Y_{n+1}^N|X_n). \tag{B.16}$$

De la même manière, on peut vérifier que tous les calculs définis sur $\mathbb{P}$ s'étendent à $\bar{\mathbb{P}}$, à condition de remplacer l'opérateur $\sum$ par l'opérateur max. Par conséquent, toutes les stratégies définies pour le critère du MPM s'étendent directement au critère du MAP.

B.4 Information extrinsèque

La notion d'information extrinsèque représente un produit intermédiaire d'un algorithme de propagation de croyance. Elle est communément utilisée pour expliquer la structure d'algorithmes itératifs. Soit X une variable aléatoire qui doit être estimée à partir de deux mesures Y et Z. On a la décomposition

$$\mathbb{P}(X|Y,Z) = \mathbb{P}(X) \cdot \frac{\mathbb{P}(Y|Z)}{\mathbb{P}(Y)} \cdot \frac{\mathbb{P}(Z|X,Y)}{\mathbb{P}(Z|Y)}. \tag{B.17}$$

Le premier terme représente *l'information a priori* sur X, le second l'information de la première mesure Y sur X et le dernier l'information restante apportée par Z sur X une fois que Y est connu, c'est à dire *l'information extrinsèque* $Ext_X(Z|Y)$. Cette dernière est souvent déterminée par

$$Ext_X(Z|Y) = \frac{\mathbb{P}(X|Y,Z)}{\mathbb{P}(X|Y)}. \tag{B.18}$$

Notons que $Ext_X(Z|Y)$ peut être considéré comme la distribution conditionnelle $\mathbb{P}(\Delta|X)$ d'une *pseudo* mesure Δ sur X. Elle joue le rôle du terme de droite dans l'équation B.2, et peut donc être vue comme un message renvoyé à X pour mettre à jour une estimée. Dans le contexte des modèles de Markov, l'information extrinsèque est souvent définie par

$$Ext_{X_n}(Y|Y_n) = Ext_{X_n}(Y_1^{n-1}, Y_{n+1}^{N}|Y_n). \tag{B.19}$$

Elle représente alors l'information complémentaire apportée par Y sur X_n une fois que la mesure locale Y_n est connue. La notion d'information extrinsèque telle qu'elle est définie par l'équation B.18 s'étend à $\bar{\mathbb{P}}$.

Annexe C

L'algorithme BCJR

Nous avons vu dans l'annexe précédente le problème général de l'estimation bayésienne. Nous présentons à présent un cas particulier d'algorithme d'estimation. L'algorithme BCJR (des noms de ses auteurs Bahl, Cocke, Jelinek et Raviv) [BCJR74] a été proposé pour traiter le problème général de l'estimation des états et des transitions d'une source de Markov observée à travers un canal discret sans mémoire. L'originalité de ce travail a été de proposer un algorithme d'estimation qui minimise la probabilité d'erreur symbole, là où les algorithmes utilisés classiquement, tel que l'algorithme de Viterbi associé aux codes convolutifs, minimisent la probabilité d'erreur des mots de code.

Considérons le système de transmission décrit par la figure C.1. La source est supposée être un processus de Markov à nombre d'états finis et à horloge discrète. Les M états distincts de la source de Markov sont indicés par l'entier m, $m = 0, 1, \ldots, M-1$. L'état de la source à l'instant t est noté S_t et les sorties associées X_t. Une séquence d'états de la source allant de t à t' est notée $S_t^{t'} = S_t, S_{t+1}, \ldots, S_{t'}$ et les sorties correspondantes sont $X_t^{t'} = X_t, X_{t+1}, \ldots, X_{t'}$.

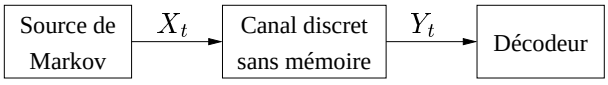


FIG. C.1 – *Schéma du système de transmission.*

Les transitions entre les états de la source de Markov dépendent des probabilités de transition

$$p_t(m|m') = \mathbb{P}(S_t = m | S_{t-1} = m'), \tag{C.1}$$

et les sorties des probabilités

$$q_t(X|m', m) = \mathbb{P}(X_t = X | S_{t-1} = m', S_t = m), \tag{C.2}$$

où X appartient à un alphabet discret fini.

La source de Markov démarre à l'état initial $S_0 = 0$ et produit une séquences de sortie X_1^T qui termine à l'état final $S_T = 0$. La séquence X_1^T est transmise sur un canal discret sans mémoire bruité, et reçue sous la forme d'une séquence $Y_1^T = Y_1, Y_2, \ldots, Y_T$. Les probabilités de transition du canal sont définies par $R(\cdot|\cdot)$ tel que pour tout $1 \leq t \leq T$

$$\mathbb{P}(Y_1^t|X_1^t) = \prod_{j=1}^{t} R(Y_j|X_j). \tag{C.3}$$

L'objectif du décodeur est d'observer Y_1^T et d'estimer les probabilités a posteriori des états et des transitions de la source de Markov, c'est à dire les probabilités conditionnelles

$$\mathbb{P}(S_t = m|Y_1^T) = \frac{\mathbb{P}(S_t = m, Y_1^T)}{\mathbb{P}(Y_1^T)} \tag{C.4}$$

et

$$\mathbb{P}(S_{t-1} = m', S_t = m|Y_1^T) = \frac{\mathbb{P}(S_{t-1} = m', S_t = m, Y_1^T)}{\mathbb{P}(Y_1^T)}. \tag{C.5}$$

Les auteurs proposent de s'appuyer sur une interprétation graphique du problème. La source de Markov peut être représentée par un graphe de transitions, comme sur l'exemple de la figure C.2-(a). Les noeuds correspondent aux états m et les branches représentent les transitions de probabilité non nulle. Si les états sont indexés à la fois par l'indice d'état m et par l'indice d'horloge t, alors un treillis est obtenu, comme sur l'exemple de la figure C.2-(b). La représentation en treillis montre la progression dans le temps de la séquence d'états. Pour chaque séquence d'états S_1^T, il existe un chemin unique dans le treillis, et réciproquement.

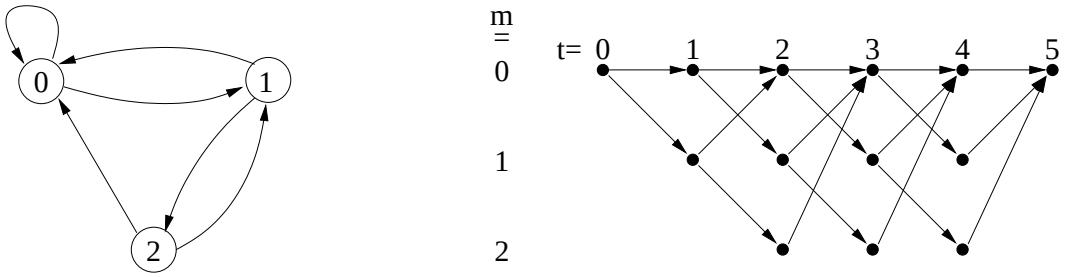


FIG. C.2 – *Exemple de source de Markov à trois états. (a) graphe des transitions entre états, (b) treillis correspondant.*

Les probabilités a posteriori $\mathbb{P}(S_t = m|Y_1^T)$ et $\mathbb{P}(S_{t-1} = m', S_t = m|Y_1^T)$ sont associées respectivement aux états et aux transitions du treillis. Pour une séquence Y_1^T donnée, $\mathbb{P}(Y_1^T)$ est une constante qui est obtenue par le décodeur. Il est alors possible

de passer par les probabilités jointes $\mathbb{P}(S_t = m, Y_1^T)$ et $\mathbb{P}(S_{t-1} = m', S_t = m, Y_1^T)$. On a

$$\begin{aligned}\mathbb{P}(S_t = m, Y_1^T) &= \mathbb{P}(S_t = m, Y_1^t) \cdot \mathbb{P}(Y_{t+1}^T | S_t = m, Y_1^t) \\ &= \mathbb{P}(S_t = m, Y_1^t) \cdot \mathbb{P}(Y_{t+1}^T | S_t = m),\end{aligned} \tag{C.6}$$

du fait de la propriété de Markov qui entraîne que si S_t est connu, les évènements d'après l'instant t ne dépendent pas de Y_1^t. De la même manière, on a

$$\mathbb{P}(S_{t-1} = m', S_t = m, Y_1^T) = \mathbb{P}(S_{t-1} = m', Y_1^{t-1}) \cdot \mathbb{P}(S_t = m, Y_t | S_{t-1} = m') \cdot \mathbb{P}(Y_{t+1}^T | S_t = m), \tag{C.7}$$

et pour $t = 1, 2, \ldots, T$

$$\begin{aligned}\mathbb{P}(S_t = m, Y_1^t) &= \sum_{m'=0}^{M-1} \mathbb{P}(S_{t-1} = m', S_t = m, Y_1^T) \\ &= \sum_{m'=0}^{M-1} \mathbb{P}(S_{t-1} = m', Y_1^{t-1}) \cdot \mathbb{P}(S_t = m, Y_t | S_{t-1} = m').\end{aligned} \tag{C.8}$$

Pour $t = 0$, on a les conditions limites $\mathbb{P}(S_t = 0, Y_1^t) = 1$ et $\mathbb{P}(S_t = m, Y_1^t) = 0$ pour $m > 0$. Pour $t = 1, 2, \ldots, T-1$, on a également

$$\begin{aligned}\mathbb{P}(Y_{t+1}^T | S_t = m) &= \sum_{m'=0}^{M-1} \mathbb{P}(S_{t+1} = m', Y_{t+1}^T | S_t = m) \\ &= \mathbb{P}(S_{t+1} = m', Y_{t+1} | S_t = m) \cdot \mathbb{P}(Y_{t+2}^T | S_{t+1} = m'),\end{aligned} \tag{C.9}$$

avec les conditions limites appropriées $\mathbb{P}(Y_T | S_t = 0) = 1$ et $\mathbb{P}(Y_T | S_t = m) = 0$ pour $m > 0$. Les relations C.8 et C.9 montrent que $\mathbb{P}(S_t = m, Y_1^t)$ et $\mathbb{P}(Y_{t+1}^T | S_t = m)$ sont calculables récursivement. Finalement, on a

$$\begin{aligned}\mathbb{P}(S_t = m, Y_t | S_{t-1} = m') &= \sum_X \mathbb{P}(S_t = m | S_{t-1} = m') \cdot \mathbb{P}(X_t = X | S_{t-1} = m', S_t = m) \cdot \mathbb{P}(Y_t | X) \\ &= \sum_X p_t(m|m') \cdot q_t(X|m', m) \cdot R(Y_t|X)\end{aligned} \tag{C.10}$$

où la somme est effectuée sur toutes les valeurs possibles des sorties X.

Le calcul de $\mathbb{P}(S_t = m, Y_1^T)$ et $\mathbb{P}(S_{t-1} = m', S_t = m, Y_1^T)$ se résume ainsi :

1. initialisation suivant les conditions limites définies ci-dessus.
2. Au fur et à mesure de la réception des observations Y_t, le décodeur calcule récursivement $\mathbb{P}(S_t = m, Y_t | S_{t-1} = m')$ et $\mathbb{P}(S_t = m, Y_1^t)$ respectivement avec les équations C.10 et C.8. Les valeurs de $\mathbb{P}(S_t = m, Y_1^t)$ sont stockées pour toutes les paires d'indices (t, m).
3. Une fois que la séquence Y_1^T a été reçue, le décodeur calcule $\mathbb{P}(Y_{t+1}^T | S_t = m)$ récursivement avec l'équation C.9. Le résultat final est obtenu grâce aux équations C.6 et C.7

Cet algorithme peut s'appliquer sur n'importe quel code, du moment qu'un treillis peut lui être associé. Les exemple des codes convolutifs et des codes en blocs sont présentés dans [BCJR74].

Publications

Brevet :

- Procédé de décodage robuste de codes arithmétiques et quasi-arithmétiques - Dépôt de brevet en cours.

Journaux :

- T. Guionnet, C. Guillemot - Error and erasure resilient compression scheme for transmission over wireless channels - en préparation.

- T. Guionnet, C. Guillemot - Soft and joint source-channel decoding of quasi-arithmetic codes - à paraître dans *EURASIP journal on Applied Signal Processing.*

- T. Guionnet, C. Guillemot - Soft decoding and synchronization of arithmetic codes : Application to image transmission over noisy channels - à paraître dans *IEEE transactions on Image Processing.*

Conférences :

- T. Guionnet, C. Guillemot - Robust decoding of arithmetic codes for image transmission over error-prone channels - dans *International Conference on Image Processing*, ICIP, 14-17 septembre 2003, Barcelone, Espagne.

- T. Guionnet, C. Guillemot, E. Fabre - Soft decoding of multiple descriptions - dans *IEEE International Conference on Multimedia*, ICME, 26-29 août 2002, Lausanne, Suisse.

- T. Guionnet, C. Guillemot, S. Pateux - Embedded multiple description coding for progressive image transmission over unreliable channels - dans *International Conference on Image Processing*, ICIP, 7-10 octobre 2001, Thessalonique, Grèce.

Bibliographie

[ABE+96] Albanese (J.), Blomer (A.), Edmonds (J.), Luby (M.) et Sudan (M.). – Priority encoded transmission. *IEEE trans. on Information Theory*, vol. 42, n° 6, Nov. 1996, pp. 1737–1744.

[Abr63] Abramson (N.). – *Information theory and coding.* – McGraw-Hill, 1963.

[Apo00] Apostolopoulos (J. G.). – Error-resilient video compression through the use of multiple states. *In : proc. ICIP'00.* – 2000.

[ASPEF01] Alasti (M.), Sayrafian-Pour (K.), Ephremides (A.) et Farvardin (N.). – Multiple description coding in networks with congestion problem. *IEEE Trans. on Information Theory*, vol. 47, n° 3, Mar. 2001, pp. 891–902.

[AW01] Apostolopoulos (J. G.) et Wee (S. J.). – Unbalanced multiple description video communication using path diversity. *In : proc. ICIP'01.* – 2001.

[BCI+97] Boyd (C.), Cleary (J. G.), Irvine (S. A.), Rinsma-Melchert (I.) et Witten (I. H.). – Integrating error detection into arithmetic coding. *IEEE Trans. on Communications*, vol. 45, n° 1, Jan. 1997, pp. 1–3.

[BCJR74] Bahl (L. R.), Cocke (J.), Jelinek (F.) et Raviv (J.). – Optimal decoding of linear codes for minimizing symbol error rates. *IEEE trans. on Information Theory*, vol. IT-20, n° 2, Mar. 1974, pp. 284–287.

[BH00a] Bauer (R.) et Hagenauer (J.). – Iterative source/channel decoding based on a trellis represent ation for variable length codes. *In : Proc. IEEE Intl. Symposium on Information Theory*, p. 117. – Piscataway, NJ, USA, June 2000.

[BH00b] Bauer (R.) et Hagenauer (J.). – Turbo fec/vlc decoding and its application to text compression. *In : Proc. Conf. on Information Science and System*, pp. WA6.6–WA6.11. – Princeton, NJ, USA, Mar. 2000.

[Bla84] Blahut (R. E.). – *Theory and Practice of Error Correcting Codes*, chap. 8. – Addison-Wesley Publishing Company, 1984.

[Bla92] Blahut (R. E.). – *Algebraic Methods for Signal Processing and Communications Coding*, chap. 4. – Springer-Verlag New York, Inc., 1992.

[BV94] Batllo (J.-C.) et Vaishampayan (V.A.). – Multiple description transform codes with an application to packetized speech. *In : proc. ISIT'94.* – 1994.

[BV97] Batllo (J.-C.) et Vaishampayan (V.A.). – Asymptotic performance of multiple description transform codes. *IEEE Trans. on Information Theory*, vol. 43, n° 2, Mar. 1997, pp. 703–707.

[BVL94] Buzi (L.), Vaishampayan (V. A.) et Laroia (R.). – Design and asymptotic performance of a structured multiple description vector quantizer. *In : proc. ISIT'94.* – 1994.

[BWR] Berger-Wolf (Tanya Y.) et Reingold (Edward M.). – Index asignment for multichannel communication under failure. *IEEE Trans. on Information Theory, to appear.*

[BWR99] Berger-Wolf (Tanya Y.) et Reingold (Edward M.). – Optimal index assignment for multichannel communication. *In : SODA.* – 1999.

[BZLS01] Boulgouris (N. V.), Zachariadis (K. E.), Leontaris (A. N.) et Strintzis (M. G.). – Drift-free multiple description coding of video. *In : proc. MMSP'01.* – 2001.

[CAK01] Channappayya (S. S.), Abousleman (G. P.) et Karam (L. J.). – Coding of digital imagery for transmission over multiple noisy channels. *In : proc. ICASSP'01.* – 2001.

[Car01] Cardinal (J.). – Design of asymmetric tree-structured multiple description source codes. *In : Proc. MMSP'01.* – 2001.

[CC00] Cao (L.) et Chen (C. W.). – Context based multiple bitstream image transmission over noisy channels. *In : proc. of SPIE Image and Video Communications and processing*, pp. 282–289. – 2000.

[CEGK98] Côté (G.), Erol (B.), Gallant (M.) et Kossentini (F.). – H263+: Video coding at low bit rates. *IEEE Trans. on circuits and systems for video technology*, vol. 8, n° 7, Nov. 1998, pp. 849–866.

[CEL02] Cardinal (J.), Elloumi (S.) et Labbé (M.). – Local optimization of index assignments for multiple description coding. *In : proc. of the 11th European Signal Processing Conference, EUSIPCO'02.* – 2002.

[CGA01] Chen (Z.), Guillemot (C.) et Ansari (R.). – Multiple description coding using multiwavelet transforms. *proc. Picture Coding Symposium, PCS'01*, 2001.

[CKS01] Coward (H.), Knopp (R.) et Servetto (S. D.). – On the performance of multiple description codes over bit error channels. *In : Proc. IEEE Intl. Symposium on Information Theory, ISIT'01.* – 2001.

[CR00] Chou (J.) et Ramchandran (K.). – Arithmetic coding-based continuous error detection for efficient arq-based image transmission. *IEEE Journal on Selected Areas in Communication*, vol. 18, n° 6, June 2000, pp. 861–867.

[CSO01] Comas (D.), Singh (R.) et Ortega (A.). – Rate-distorsion optimization in a robust video transmission based on unbalanced multiple description coding. *In : proc. MMSP'01.* – 2001.

[CTK01] Cammas (N.), Teniou (G.) et Kervadec (S.). – *Codage par descriptions multiples dans un codeur JPEG2000.* – Rapport technique, IFSIC, Université de Rennes 1, Jan. 2001.

[CW98] Chung (D.) et Wang (Y.). – Multiple description image coding using signal decomposition and reconstruction based on lapped orthogonal transforms. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'98.* – 1998.

[CW00] Chung (D.) et Wang (Y.). – Lapped orthogonal transform designed for error resilient image coding. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'00.* – 2000.

[DHS01] Demiroglu (C.), Hoffman (M. W.) et Sayood (K.). – Joint source channel coding using arithmetic codes and trellis coded modulaion. *In : Proc. Data Compression Conf.*, pp. 302–311. – Snowbird, UT, USA, Mar. 2001.

[DS98] Demir (N.) et Sayood (K.). – Joint source-channel coding for variable length codes. *In : Proc. IEEE Data Compression Conf.*, pp. 139–148. – Snowbird, UT, USA, Mar. 1998.

[DSV00] Dragotti (P. L.), Servetto (S. D.) et Vetterli (M.). – Analysis of optimal filter banks for multiple description coding. *Proc. IEEE Data Compression Conf., DCC'00*, 2000, pp. 323–332.

[DSV02] Dragotti (P. L.), Servetto (S. D.) et Vetterli (M.). – Optimal filter banks for multiple description coding : analysis and synthesis. *IEEE Trans. on Information Theory*, vol. 48, n° 7, Jul. 2002, pp. 2036–2451.

[EC91] Equitz (W. H. R.) et Cover (T. M.). – Successive refinement of information. *IEEE Trans. on Information Theory*, vol. 37, n° 2, Mar. 1991, pp. 269–275.

[Elm99a] Elmasry (G. F.). – Embedding channel coding in arithmetic coding. *IEE Proc.-Communications*, vol. 146, n° 2, Apr. 1999, pp. 73–78.

[Elm99b] Elmasry (G. F.). – Joint lossless-source and channel coding using automatic repeat request. *IEEE Trans. on Communications*, vol. 47, n° 7, July 1999, pp. 953–955.

[Eve63] Everett (H.). – Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Oper. Res.*, vol. 11, 1963, pp. 399–417.

[FDZ00] Frank-Dayan (Y.) et Zamir (R.). – Universal lattice-based quantizers for multiple descriptions. *In : Proc. IEEE Intl. Conf. on Data Compression, DCC'00.* – 2000.

[FE99] Fleming (M.) et Effros (M.). – Generalized multiple description vector quantization. *In : Proc. IEEE Intl. Conf. on Data Compression, DCC'99.* – March 1999.

[FFL02] Franchi (N.), Fumagalli (M.) et Lancini (R.). – Flexible redundancy insertion in a polyphase downsampling multiple description image coding. *In : proc. ICME'02.* – 2002.

[Gab01] Gabay (A.). – *Codage conjoint source canal: Application aux transmissions d'images par satellite.* – Thèse de PhD, Ecole Nationale Supérieure des Télécommunications, Janvier 2001.

[GC82] Gamal (A.A. El) et Cover (T.M.). – Achievable rates for multiple descriptions. *IEEE Trans. on Information Theory*, vol. IT-28, n° 6, Nov. 1982, pp. 851–857.

[GCSR95] Gelman (A.), Carlin (J. B.), Stern (H. S.) et Rubin (D. B.). – *Bayesian Data Analysis.* – Texts in Statistical Science, Chapman & Hall, 1995.

[GDR00] Gabay (A.), Duhamel (P.) et Rioul (O.). – Spectral interpolation coder for impulse noise cancellation over a binary symetric channel. *In : EUSIPCO 2000.* – Tampere, Finland, September 2000.

[GFGR01] Guyader (A.), Fabre (E.), Guillemot (C.) et Robert (M.). – Joint source-channel turbo decoding of entropy coded sources. *IEEE Journal on Selected Areas in Communication, special issue on the turbo principle : from theory to practice*, vol. 19, n° 9, Sept. 2001, pp. 1680–1696.

[GK98] Goyal (V.K.) et Kovacevic (J.). – Optimal multiple description transform coding of gaussian vectors. *Proc. IEEE Data Compression Conf., DCC'98*, Mar. 1998, pp. 388–397.

[GKAV98] Goyal (V.K.), Kovacevic (J.), Arean (R.) et Vetterli (M.). – Multiple description transform coding of images. *Proc. IEEE Intl. Conf. on Image Processing, ICIP'98*, Oct. 1998, pp. 674–678.

[GKV98] Goyal (V.K.), Kovacevic (J.) et Vetterli (M.). – Multiple description transform coding: Robustness to erasures using tight frame expansions. *Proc. IEEE Intl. Symposium on Information Theory*, Aug. 1998.

[Goy01] Goyal (V. K.). – Multiple description coding : compression meets the network. *IEEE Signal Processing Magazine*, vol. 18, n° 5, Sep. 2001, pp. 74–93.

[GSK01] Gallant (M.), Shirani (S.) et Kossentini (F.). – Standard-compliant multiple description video coding. *In : proc. ICIP'01.* – 2001.

[Guy02] Guyader (A.). – *Contribution aux algorithmes de décodage pour les codes graphiques.* – Thèse de PhD, Université de Rennes 1, 2002.

[Hem96] Hemami (S. S.). – Reconstruction-optimized lapped orthogonal transforms for robust image transmission. *IEEE Trans. on Circuits and systems for Video Technology*, vol. 6, n° 2, Apr. 1996, pp. 168–181.

[Hén02] Hénocq (Xavier). – *Contrôle d'erreur pour transmission de flux vidéo temps réel sur réseaux de paquets hétérogènes et variant dans le temps.* – Thèse de PhD, Université de Rennes 1, 2002.

[HSHW95] Hardman (V.), Sasse (M. A.), Handley (M.) et Watson (A.). – Reliable audio for use over the internet. *In : proc. INET'95.* – 1995.

[HV92a] Howard (P. G.) et Vitter (J. S.). – Analysis of arithmetic coding for data compression. *Information Processing and Management*, vol. 28, n° 6, Nov.-Dec. 1992, pp. 749–763.

[HV92b] Howard (P. G.) et Vitter (J. S.). – *Image and Text Compression*, pp. 85–112. – Kluwer Academic Publisher, 1992.

[HV93] Howard (P. G.) et Vitter (J. S.). – Design and analysis of fast text compression based on quasi-arithmetic coding. *In : Data Compression Conference*, pp. 98–107. – Snowbird, Utah, March-April 1993.

[HV94] Howard (P. G.) et Vitter (J. S.). – Arithmetic coding for data compression. *Proc. IEEE*, vol. 82, n° 6, 1994, pp. 857–865.

[ISO00] ISO/IEC 15444-1:2000. – *JPEG 2000 image coding system – Part 1*, Dec. 2000, first édition.

[ISO02] ISO/IEC JTC1/SC29/WG11 N4668. – *Overview of the MPEG-4 Standard*, March 2002.

[JO99] Jiang (W.) et Ortega (A.). – Multiple description coding via polyphase transform and selective quantization. *In : Proc. SPIE Intl. Conf. on Visual Communications and Image Processing, VCIP*, pp. 998–1008. – Jan. 1999.

[JO02] Jiang (W.) et Ortega (A.). – Multiple description speech coding for robust communication over lossy packet networks. *In : proc. ICME'02.* – 2002.

[Joi02] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG. – *Editor's proposed draft text modification for joint video specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)*, Aug. 2002.

[JT99] Jafarkhani (H.) et Tarokh (V.). – Multiple description trellis-coded quantization. *IEEE Trans. on Communications*, vol. 47, n° 6, Jun. 1999, pp. 799–803.

[KCR98] Kozintsev (I.), Chou (J.) et Ramchandran (K.). – Image transmission using arithmetic coding based continuous error detection. *In : Proc. Data Compression Conf.*, pp. 339–348. – Snowbird, UT, USA, Mar. 1998.

[KGK00] Kelner (J. A.), Goyal (V. K.) et Kovacevic (J.). – Multiple description lattice vector quantization variations and extensions. *In : Proc. IEEE Intl. Conf. on Data Compression, DCC'00.* – 2000.

[LAK00] Lam (T.-T.), Abousleman (G. P.) et Karam (L. J.). – Image coding with robust channel-optimized trellis coded quantization. *IEEE Journal on Selected Areas in Communications*, Jun. 2000, pp. 940–951.

[LAM02] Lee (Y-C), Altunbasak (Y.) et Mersereau (R. M.). – A two-stage multiple description video coder with drift-preventing motion compensated prediction. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'02.* – 2002.

[Lan84] Langdon (G. G.). – An introduction to arithmetic coding. *IBM J. Res. Develop.*, vol. 28, n° 2, Mar. 1984, pp. 135–149.

[Lau96] Lauritzen (S. L.). – *Graphical Models.* – Oxford Statistical Science Series 17, Oxford University Press, 1996.

[LR92] Lam (W.M.) et Reibman (A.R.). – Self-synchronizing variable-length codes for image transmission. *In : Proc. IEEE Intl. Conf. on Accoustic Speech and Signal Processing*, pp. 477–480. – New York, NY, USA, Sept. 1992.

[LSG01] Liang (Y. J.), Steinbach (E. G.) et Girod (B.). – Multi-stream voice over ip using packet path diversity. *In : proc. MMSP'01.* – 2001.

[LT01] Lawabni (A. E.) et Tewfik (A. H.). – Image transmission over error-prone channels : sigma filtering and multiple description objectives. *In : proc. MMSP'01.* – 2001.

[LV98a] Lebrun (J.) et Vetterli (M.). – Balanced multiwavelets theory and design. *IEEE Transactions on Signal Processing, Special issue on Wavelets*, vol. 46, n° 4, Apr. 1998, pp. 1119–1125.

[LV98b] Lebrun (J.) et Vetterli (M.). – High order balanced multiwavelets. *proc. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98*, May 1998, pp. 1529–1532.

[MBHW01] Marpe (D.), Blättermann (G.), Heising (G.) et Wiegand (T.). – Video compression using context-based adaptive arithmetic coding. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'01.* – Thessaloniki, Gr, Oct. 2001.

[MF90] Marcelli (M. W.) et Fischer (T. R.). – Trellis coded quantization of memoryless and gauss-markov sources. *IEEE Trans. on Communications*, vol. 38, n° 1, Jan. 1990, pp. 82–93.

[MF98] Murad (A.H.) et Fuja (T.E.). – Joint source-channel decoding of variable length encoded sources. *In : Proc. Information Theory Workshop*, pp. 94–95. – New York, NY, USA, June 1998.

[MGBB00] Marcellin (M.W.), Gormish (M.J.), Bilgin (A.) et Boliek (M.P.). – An overview of jpeg-2000. *In : Proc. Data Compression Conf.*, pp. 523–541. – Snowbird, UT, USA, Mar. 2000.

[MHET99] Marvasti (F.), Hasan (M.), Echhart (M.) et Talebi (S.). – Efficient algorithms for burst error recovery using FFT and other transform kernels. *IEEE Transactions on Signal Processing*, vol. 47(4), April 1999, pp. 1065–1075.

[MMR99] Miguel (A. C.), Mohr (A. E.) et Riskin (E. A.). – Spiht for generalized multiple description coding. *Proc. IEEE Intl. Conf. on Image Processing, ICIP'99*, 1999.

[MP98] Miller (D. J.) et Park (M.). – A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden markov models. *IEEE Trans. on Communications*, vol. 46, n° 2, Feb. 1998, pp. 222–231.

[MR00] Miguel (A. C.) et Riskin (E. A.). – Protection of regions of interest against data loss in generalized multiple description framework. *Proc. IEEE Data Compression Conf., DCC'00*, 2000.

[MRL99a] Mohr (A. E.), Riskin (E. A.) et Ladner (R. E.). – Generalized multiple description coding through unequal loss protection. *Proc. IEEE Intl. Conf. on Image Processing, ICIP'99*, 1999.

[MRL99b] Mohr (A. E.), Riskin (E. A.) et Ladner (R. E.). – Graceful degradation over packet erasure channels through forward error correction. *Proc. IEEE Data Compression Conf., DCC'99*, Mar. 1999, pp. 92–101.

[MS89] Malvar (H. S.) et Staelin (D. H.). – The lot : transform coding without blocking effects. *IEEE Trans. on Accoustics, Speech and Signal Processing*, vol. 38, n° 4, Apr. 1989, pp. 553–559.

[NZ01] Nathan (R.) et Zamir (R.). – Multiple description video coding with unquantized prediction loop. *In : proc. ICIP'01.* – 2001.

[OWVR97] Orchard (M.T.), Wang (Y.), Vaishampayan (V.A.) et Reibman (A.R.). – Redundancy rate-distortion analysis of multiple description coding using pairwise correlating transforms. *Proc. IEEE Intl. Conf. on Image Processing, ICIP'97*, vol. I, Oct. 1997, pp. 608–611.

[Oza80] Ozarow (L.). – On a source coding problem with two channels and three receivers. *Bell Syst. Tech. J.*, vol. 59, n° 10, Dec. 1980, pp. 1909–1921.

[PA00] P.Sagetong et A.Ortega. – Optimal bit allocation for channel-adaptive multiple description coding. *In : proc. Image and Video Communication and Processing*, pp. 53–63. – Jan. 2000.

[PAB02] Pereira (M.), Antonini (M.) et Barlaud (M.). – Channel adapted scan-based multiple description video coding. *In : proc. ICME'02.* – 2002.

[Pap91] Papoulis (A.). – *Probability, Random Variables, and Stochastic Processes*, pp. 552, 562. – McGraw-Hill International Editions, 1991.

[Pas76] Pasco (R.). – *Source coding algorithms for fast data compression.* – Thèse de PhD, Dept. of Electrical Engineering, Stanford Univ., Stanford Calif., 1976.

[PHS01] Pettijohn (B. D.), Hoffman (M. W.) et Sayood (K.). – Joint source/channel coding using arithmetic codes. *IEEE Trans. on Communications*, vol. 49, n° 5, May 2001, pp. 826–836.

[PM98] Park (M.) et Miller (D.J.). – Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs. *In : Proc. Conf. on Information Sciences and Systems*, pp. 477–482. – Princeton, NJ, USA, May 1998.

[PMLA88] Pennebaker (W. B.), Mitchell (J. L.), Langdon (G. G.) et Arps (R. B.). – An overview of the basic principles of the q-coder adaptive binary arithmetic coder. *IBM J. Res. Develop.*, vol. 32, n° 6, Nov. 1988, pp. 717–726.

[PPR02a] Puri (R.), Pradhan (S.) et Ramchandran (K.). – n-channel multiple descriptions: Theory and constructions. *In : Proc. IEEE Data Compression Conf., DCC'02*, pp. 263–271. – 2002.

[PPR02b] Puri (R.), Pradhan (S.) et Ramchandran (K.). – n-channel symmetric multiple descriptions: New rate regions. *In : Proc. IEEE Intl. Symposium on Information Theory, ISIT 2002.* – Lausanne, Switzerland, Jun-Jul 2002.

[PR99] Puri (R.) et Ramchandran (K.). – Multiple description source coding using forward error correction codes. *Proc. 33d Asilomar Conference on Signals, System and Computers*, Oct. 1999.

[Ris76] Rissanen (J. J.). – Generalized kraft inequality and arithmetic coding. *IBM J. Res. Develop.*, vol. 20, May 1976, pp. 198–203.

[Ris79] Rissanen (J. J.). – Arithmetic codings as number representations. *Acta Polytech. Scand. Math.*, vol. 31, Dec. 1979, pp. 44–51.

[RJW+99] Reibman (A. R.), Jafarkhani (H.), Wang (Y.), Orchard (M. T.) et Puri (R.). – Multiple description coding for video using motion compensated prediction. *In : IEEE International Conference on Image Processing, ICIP'99.* – Kobe, Japan, october 1999.

[RJWO01] Reibman (A.), Jafarkhani (H.), Wang (Y.) et Orchard (M.). – Multiple description video using rate-distorsion splitting. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'01.* – Thessaloniki, Gr, Oct. 2001.

[Roz99] Rozic (N.). – Image/video communications: Joint source/channel coding. *International Journal Of Communication Systems*, vol. 12, 1999, pp. 23–47.

[RPJ+99] Reibman (A. R.), Puri (R.), Jafarkhani (H.), Wang (Y.) et Orchard (M. T.). – *Multiple Description Video Coding Using Motion Compensated Prediction.* – Rapport technique, Berlin, Proposal H26L : ITU 6Telecommunications Standardization Sector, Video Coding Experts Group (Question 15), eighth meeting,, August 1999.

[RR00] Regunathan (S. L.) et Rose (K.). – Efficient prediction in multiple description video coding. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'00.* – 2000.

[RV00] Ramac (L. C.) et Varshney (P. K.). – A wavelet domain diversity method for transmission of images over wireless channels. *IEEE Journal on Selected Areas in Communication*, vol. 18, n° 6, Jun. 2000, pp. 891–898.

[SAR00] Sachs (D. G.), Anand (R.) et Ramchandran (K.). – Wireless image transmission using multiple-description based concatenated codes. *Proc. IEEE Data Compression Conf., DCC'00*, 2000.

[Say99] Sayir (J.). – *On coding by probability transformation.* – Thèse de PhD, Swiis Federal Institute of Technology, Zurich, 1999.

[Say00] Sayood (Khalid). – *Introduction to data compression.* – Morgan Kaufmann Publishers, 2000.

[SC98] Srinivasan (M.) et Chellappa (R.). – Multiple description subband coding. *Proc. IEEE Intl. Conf. on Image Processing, ICIP'98*, Oct. 1998, pp. 684–688.

[SCW00] Sodagar (I.), Chai (B. B.) et Wus (J.). – A new error resilience technique for image compression using arithmetic coding. *In : IEEE Intl. Conf. on Accoustics, Speech and Signal Processing*, pp. 2127–2130. – Istanbul, Turkey, June 2000.

[Sha48] Shannon (C. E.). – A mathematical theory of communication. *Bell Syst. Tech. J.*, vol. 27, July 1948, pp. 398–403.

[Sha59] Shannon (C. E.). – Coding theoreme for a discrete source with a fidelity criterium. *IRE Nat. Conv. Rec., Reprinted in "Key Papers in the Development of Information Theory. New York: IEEE*, 1959.

[SJJ99] S-J.Choi et J.Woods. – Motion-Compensated 3-D Subband Coding of Video. *IEEE Trans. on Image Processing*, vol. 8, n° 2, février 1999, pp. 155–167.

[SN99] Servetto (S. D.) et Nahrstedt (K.). – Video streaming over the public internet: Multiple description codes and adaptive transport protocols. *In : IEEE International Conference on Image Processing, ICIP'99.* – Kobe, Japan, october 1999.

[SOPJ00] Singh (R.), Ortega (A.), Perret (L.) et Jiang (W.). – Comparison of multiple description coding and layered coding based on network simulations.

In : Image and Video Communications and Processing, proc. SPIE Vol. 3974, pp. 929–939. – 2000.

[SP96] Said (Amir) et Pearlman (William A.). – A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on circuits and systems for video technology*, vol. 6, n° 3, June 1996, pp. 243–250.

[Sri99] Srinivasan (M.). – Iterative decoding of multiple descriptions. *In : Proc. IEEE Intl. Conf. on Data Compression, DCC'99.* – March 1999.

[SRVN00] Servetto (S.D.), Ramchandran (K.), Vaishampayan (V.A.) et Nahrstedt (K.). – Multiple description wavelet based image coding. *IEEE Trans. on Image Processing*, vol. 9, n° 5, May 2000, pp. 813–826.

[SV01] Subbalakshmi (K. P.) et Vaisey (J.). – On the joint source-channel decoding of variable-length encoded sources: The bsc case. *IEEE Trans. on Communications*, vol. 49, n° 12, Dec. 2001, pp. 2052–2055.

[SVS99] Servetto (S. D.), Vaishampayan (V. A.) et Sloane (N. J. A.). – Multiple description lattice vector quantization. *In : Proc. IEEE Intl. Conf. on Data Compression, DCC'99.* – March 1999.

[Tau00] Taubman (D.). – High performance scalable image compression with ebcot. *IEEE Trans. on Image Processing*, vol. 9, n° 7, July 2000, pp. 1158–1170.

[TF84] T.J. Ferguson (J.H. Rabinowitz). – Self-synchronizing huffman codes. *IEEE Trans. on Information Theory*, vol. IT-30, n° 4, July 1984, pp. 687–693.

[TY01] Tanaka (T.) et Yamashita (Y.). – The orientation adaptive lapped biorthogonal transform for efficient image coding. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'01.* – 2001.

[TZ94] Taubman (D.) et Zakhor (A.). – Multi-rate 3-d subband coding of video. *IEEE Trans. on Image Processing*, vol. 3, n° 5, Sep. 1994, pp. 572–588.

[TZ99] Tan (W-T) et Zakhor (A.). – Real-time internet video using error resilient scalable compression and tcp-friendly transport protocol. *IEEE Trans. on Multimedia*, vol. 1, n° 2, Jun. 1999, pp. 172–186.

[TZ01] Tang (X.) et Zakhor (A.). – Matching pursuits multiple description coding for wireless video. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'01.* – Thessaloniki, Gr, Oct. 2001.

[Vai93] Vaishampayan (V.A.). – Design of multiple description scalar quantizers. *IEEE Trans. on Information Theory*, vol. 39, n° 3, May 1993, pp. 821–834.

[VB98] Vaishampayan (V.A.) et Batllo (J.-C.). – Asymptotic analysis of multiple description quantizers. *IEEE Trans. on Information Theory*, vol. 44, n° 1, Jan. 1998, pp. 278–284.

[VBC98] Vaishampayan (V.A.), Batllo (J.-C.) et Calderbank (A.R.). – On reducing granular distorsion in multiple description quantization. *In : proc. ISIT'98.* – 1998.

[VD94] Vaishampayan (V.A.) et Domaszewicz (J.). – Design of entropy constrained multiple description scalar quantizers. *IEEE Trans. on Information Theory*, vol. 40, Jan. 1994, pp. 245–251.

[VFE00] Varnica (N.), Fleming (M.) et Effros (M.). – Multi-resolution adaptation of the spiht algorithm for multiple description. *Proc. IEEE Data Compression Conf., DCC'00*, 2000.

[VJ99] Vaishampayan (V. A.) et John (S.). – Balanced interframe multiple description video compression. *In : proc. ICIP'99.* – 1999.

[VK79] Viterbi (A. J.) et K. (Omura J.). – *Principles of digital communication and coding.* – McGraw-Hill series in electrical engineering, 1979.

[VSS01] Vaishampayan (V. A.), Sloane (N. J. A.) et Servetto (S. D.). – Multiple description vector quantization with lattice codebooks : design and analysis. *IEEE trans. on Information Theory*, vol. 47, n° 5, Jul. 2001, pp. 1718–1734.

[WL01] Wang (Y.) et Lin (S.). – Error resilient video coding using multiple description motion compensation. *In : proc. MMSP'01.* – 2001.

[WNC87] Witten (I. H.), Neal (R. M.) et Cleary (J. G.). – Arithmetic coding for data compression. *Communications of the ACM*, vol. 30, n° 6, June 1987, pp. 520–540.

[WNC98] Witten (I. H.), Neal (R. M.) et Cleary (J. G.). – Arithmetic coding revisited. *ACM Trans. on Information Systems*, vol. 16, n° 3, July 1998, pp. 256–294.

[WO00] Wang (X.) et Orchard (M. T.). – Multiple description coding using trellis coded quantization. *In : proc. ICIP'00.* – 2000.

[WOR98] Wang (Y.), Orchard (M.) et Reibman (A.). – Optimal pairwise correlating transforms for multiple description coding. *In : IEEE International Conference on Image Processing (ICIP98).* – Chicago, Illinois, October 1998.

[WS95] Wu (J.) et Shiu (J.). – Discrete cosine transform in error control coding. *IEEE Transactions on communications*, vol. 43(5), may 1995, pp. 1857–1861.

[WV98] Wen (J.) et Villasenor (J. D.). – Reversible variable length codes for efficient and robust image and video coding. *In : Proc. IEEE Data Compression Conf.*, pp. 471–480. – Snowbird, UT, USA, Apr. 1998.

[WWZ80] Wolf (J. K.), Wyner (A. D.) et Ziv (J.). – Source coding for multiple descriptions. *Bell Syst. Tech. J.*, vol. 59, n° 8, Oct. 1980, pp. 1417–1426.

[YR98] Yang (X.) et Ramchandran (K.). – Optimal multiple description subband coding. *In : Proc. IEEE Intl. Conf. on Image Processing, ICIP'98*, pp. 654–658. – Chicago, Oct. 1998.

[YR00] Yang (X.) et Ramchandran (K.). – Optimal subband filter banks for multiple description coding. *IEEE Trans. on Information Theory*, vol. 46, n° 7, Nov. 2000, pp. 2477–2490.

[YTM95] Y. Takishima (M. Wada) et Murakami (H.). – Reversible variable length codes. *IEEE Trans. on Communications*, vol. 43, n° 4, Apr. 1995, pp. 158–162.

[YV95] Yang (S-M.) et Vaishampayan (V.A.). – Low-delay communication for rayleigh fading channels. *IEEE Trans. on Communications*, vol. 43, n° 11, Nov. 1995, pp. 2771–2783.

[ZG90] Zeger (K.) et Gersho (A.). – Pseudo-gray coding. *IEEE trans. on Communications*, vol. 38, n° 12, Dec. 1990, pp. 2147–2158.

[ZMG02] Zhang (Yi), Motani (M.) et Garg (H. K.). – Wireless video transmission using multiple description codes combined with prioritized dct compression. *In : proc. of ICME, International Conference on Multimedia and Expo.* – Aug 2002.

Abstract

This thesis deals with multimedia data transmission over heterogeneous and time varying packet networks. We start by studying multiple description coding in order to define coding schemes which are robust to both erasures and errors, and which adapts easily to channel characteristics. First, we apply multiple description scalar quantization (MDSQ) to progressive encoding of still images with two descriptions. An algorithm is proposed to design MDSQ index assignment tables adapted to progressive encoding. The method is integrated into a JPEG2000 coder. The two descriptions image coder is robust to erasures. However, the use of arithmetic coding makes it particularly sensitive to errors. A modelisation of arithmetic coding based on bayesian networks allows us to design a soft decoding procedure. The estimation reliability can be increased by redundancy addition in a form specifically adapted to the treatment of arithmetic coding synchronisation. The algorithm is integrated into a JPEG2000 coder. Due to the structure of arithmetic coding, the soft decoding algorithm suffers from an exponential complexity. A pruning strategy allows to control the complexity, at the cost of sub-optimality of the estimation. Simplified arithmetic codes, referred to as quasi-arithmetic codes, are considered to solve this problem. The properties of these codes together with an appropriate modeling of the source allow to represent the decoder by a finite state machine. An optimal soft decoding algorithm is then proposed. Soft decoding of quasi-arithmetic codes can be combined to convolutive coding in an iterative scheme similar to serial turbo codes. Finally, MDSQ is combined to soft decoding of quasi-arithmetic codes in order to design a coding scheme which is robust to both erasures and errors, and which adapts easily to channel characteristics.

Key words

Multiple description coding, joint source and channel coding, arithmetic coding, quasi-arithmetic coding, soft decoding, bayesian networks, estimation, MAP, BCJR, JPEG2000.

Résumé

L'étude menée dans cette thèse s'inscrit dans le contexte général de la transmission de données multimédia sur des réseaux de paquets hétérogènes aux caractéristiques variant dans le temps. En prenant comme point de départ l'étude des techniques de codage par descriptions multiples, nous nous sommes attachés à définir des schémas de codage robustes à la fois aux erreurs et aux effacements et capables de s'adapter aux caractéristiques du canal. Dans un premier temps, nous appliquons la quantification scalaire à descriptions multiples (MDSQ) au codage d'image fixe à deux descriptions progressives. Nous proposons un algorithme qui permet la création de tables d'index MDSQ adaptées au codage progressif des descriptions. La méthode est intégrée dans un codeur JPEG2000. Le codeur d'image à deux descriptions est robuste face aux effacements. En revanche l'utilisation du codage arithmétique le rend particulièrement sensible aux erreurs. Une modélisation du codage arithmétique basée sur les réseaux bayésiens permet de concevoir un algorithme de décodage souple. La fiabilité de l'estimation peut être renforcée en ajoutant de la redondance sous une forme spécifiquement adaptée au traitement de la synchronisation des codes arithmétiques. L'algorithme est intégré dans un codeur JPEG2000. De par la structure des codes arithmétiques, l'algorithme de décodage souple souffre d'une complexité exponentielle. Une stratégie d'élagage permet de contrôler la complexité, mais rend l'estimation sous-optimale. Des codes arithmétiques simplifiés appelés codes quasi-arithmétiques sont considérés pour résoudre ce problème. Les propriétés de ces codes associées à une modélisation appropriée de la source permettent de représenter le décodeur sous forme d'automate à nombre d'états finis. Un nouvel algorithme de décodage souple, optimal cette fois, est développé. Le décodage souple de codes quasi-arithmétiques peut être associé à un code convolutif dans un schéma itératif inspiré des turbo codes en série. Finalement, la MDSQ est associée au décodage souple de codes quasi-arithmétiques pour proposer un schéma général de codage robuste à la fois aux effacements et aux erreurs et adaptable aux caractéristiques du canal.

Mots clés

Codage par descriptions multiples, codage conjoint source/canal, codage arithmétique, codage quasi-arithmétique, décodage souple, réseaux bayésiens, estimation, MAP, BCJR, JPEG2000.