Robust codes and joint source-channel codes for multimedia transmission over mobile channels

Hervé Jégou

N° d'ordre: 03217

THÈSE

présentée

devant l'Université de Rennes 1

pour obtenir

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

par

Hervé JÉGOU

Équipe d'accueil : Temics - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Codes robustes et codes joints source-canal pour transmission multimédia sur canaux mobiles

Soutenue le 23 novembre 2005 devant la commission d'examen

M. :	Albert	BENVENISTE	Président
MM. :	Robert M.	GRAY	Rapporteurs
	Luc	VANDENDORPE	
MM. :	Ramesh	PYNDIAH	Examinateurs
	Michel	KIEFFER	
	Christine	GUILLEMOT	Directeur de thèse

à ma famille

Remerciements

Une thèse n'est pas seulement l'aboutissement du travail d'un doctorant. C'est également une charge pour le jury et les proches. Cette courte page de remerciements leur est dédiée.

Je remercie en premier lieu Albert BENVÉNISTE pour l'honneur qu'il m'a fait de présider mon jury de thèse.

Je remercie mes rapporteurs, Robert M. GRAY, Professeur à l'Université de Stanford, et Luc VANDENDORPE, Professeur à l'Université Catholique de Louvain, pour le soin qu'ils ont apporté à la lecture et à l'évaluation de ce manuscrit.

J'associe pleinement à ces remerciements Ramesh PYNDIAH, Professeur à l'École nationale Supérieure des Télécommunications de Bretagne et Michel KIEFFER, Maître de conférence à l'Université de Paris XI, pour le remarquable travail de relecture qu'ils ont effectué en tant qu'examinateurs.

Je remercie particulièrement Christine GUILLEMOT pour m'avoir encadré, mais également formé et motivé. Je tiens à souligner son professionnalisme et ses qualités d'écoute, qui ont contribué à ce que j'estime avoir été une collaboration fructueuse. Christine, je t'exprime ma reconnaissance.

Je tiens à remercier les membres de l'équipe TEMICS, qui ont contribué à une bonne ambiance de travail durant ces trois années. Un grand merci en particulier à Jonathan, Simon et au grand Vivien.

Je remercie les membres de l'IRISA pour les discussions enrichissantes que j'ai pu avoir avec eux.

Je remercie enfin mes amis et ma famille, pour avoir supporté et accepté mes choix. Merci en particulier à Catherine, qui a pris part malgré elle à cette aventure.

Table des matières

Première partie

Problématique et résumé des contributions

Introduction

L'engouement pour la téléphonie mobile illustre l'intérêt du grand public pour les communications numériques. Plus généralement, les domaines liés au traitement de l'information présentent des enjeux économiques importants car ils correspondent à des besoins applicatifs. Pour le chercheur, ils soulèvent un certain nombre de défis pratiques et théoriques.

L'application type que nous allons considérer dans cette thèse est la transmission de sons, d'images ou de vidéos vers un terminal mobile à faible capacité de calcul. Un tel terminal est sujet à des contraintes énergétiques importantes. L'importance de l'énergie dans les problèmes de communication est fondamentale et justifie que les performances soient representées en fonction de la quantité d'énergie requise pour transmettre une unité d'information sur un canal. Cette énergie revet deux formes. D'une part, la transmission sur le canal physique utilise de l'énergie. C'est la fonction de coût usuellement considérée en communication. Par ailleurs, les calculs algorithmiques liés au traitement de l'information requièrent également de l'énergie. Les calculs s'effectuent donc au détriment de l'autonomie du terminal. Sous ces deux contraintes énergétiques, les systèmes de communications multimédia ont pour vocation de transmettre au mieux, c'est-à-dire pour une mesure de distorsion donnée, la plus grande quantité d'information possible. Pour cela, nous proposons des techniques de codage de source et de codage conjoint qui exploitent différentes hypothèses sur la source ou qui sont motivées par des raisons théoriques.

Cette thèse se compose de deux parties. La première est rédigée en français et inclut les deux premiers chapitres. Elle présente les notions de base de théorie de l'information et positionne les apports de cette thèse par rapport à l'état de l'art en codage conjoint source/canal. Elle inclut succintement quelques perspectives de recherche. La seconde partie est rédigée en anglais et est constituée des contributions. Elle contient cinq chapitres (3 à 7) qui correspondent chacun à un ou plusieurs articles de revue.

Nous proposons dans cette thèse des méthodes pour compresser et transmettre de l'information multimédia en présence d'erreurs et avec une capacité de calcul réduite. Toutes les techniques que nous proposons se situent au niveau du codage entropique, dont les fondements sont rapidement rappelés dans le chapitre 1. En effet, les systèmes de compression de signal utilisent très largement les codes à longueur variable

(Codes d'Huffman ou codes arithmétiques). Ils approchent l'entropie du signal mais sont en revanche très sensibles au bruit de transmission : une simple erreur peut ainsi conduire à une information décodée inutilisable.

Comme nous l'avons indiqué, l'apparition des applications multimédia sans fils et mobiles a fortement motivé l'étude de codes ayant une meilleure robustesse aux erreurs dans un environnement bruité. Les approches que nous allons considérer pour cela sont dites de codage de source robuste et de codage conjoint source/canal. Ces notions sont explicitées dans le chapitre 2 et nous en proposons une classification, avant de poursuivre par un résumé des contributions.

Dans le chapitre 3 de cette thèse, nous allons considérer une technique de codage entropique qui améliore nettement la robustesse pour un coût de compression dérisoire, voire nul. Cette technique vise à approcher au mieux l'entropie du signal considéré en étant plus robuste au bruit de transmission que les techniques usuelles. Elle repose sur l'utilisation de codes dits *multiplexés*. En considérant deux sources, ces codes offrent une résistance inégale aux erreurs au niveau du codage de source, celui-ci étant généralement effectué du côté canal. Plus précisement, deux sources sont considérées : une source prioritaire, i.e. qui contient la plus grande partie de l'énergie du signal, et une source moins prioritaire, qui améliore la reconstruction issue de la première source. Le codage entropique est effectué conjointement pour les deux sources, de manière à offrir à la source prioritaire une synchronisation parfaite entre les horloges bits et symboles. La technique est étendue aux sources markoviennes dans le chapitre 4. Dans ce chapitre, nous montrons également que la prise en compte des caractéristiques du canal améliore le décodage en considérant le problème de l'étiquetage binaire (index assignement, IA). Diverses techniques d'optimisation sont proposées pour effectuer cet optimisation de l'étiquetage binaire, dont le recuit simulé.

Dans le chapitre 5, nous considérons le problème de décodage souple des codes à longueur variable lorsqu'une contrainte de terminaison est disponible au décodeur. Les principales contributions de ce chapitre sont les suivantes. Premièrement, un lien est établi entre les propriétés de resynchronisation et les performances en décodage souple des codes à longueur variable. Deuxièmement, une méthode est proposée pour réduire substantiellement la complexité en agrégant les états du treillis sans que la précision de l'estimation ne soit affectée. Ce résultat est motivé par une analyse et observé en simulations.

Nous considérons également le problème de la mise en paquets des codes à longueur variables dans le chapitre 6. Nous proposons des alternatives à la concaténation des mots de code qui améliorent les performances de résistance aux erreurs ou de progressivité. Enfin, dans le chapitre 7, nous considérons des structures qui offrent un plus grand nombre de degrés de liberté pour la construction de codes à longueur variable. Ces structures se basent sur des ensembles de règles de ré-écriture. La souplesse de construction qu'ils offrent est illustrée par deux méthodes de construction offrant des propriétés impossibles à obtenir avec des codes à longueur variable classiques.

Chapitre 1

Préliminaires

Dans cette partie, nous nous proposons d'aborder quelques notions qui seront couramment utilisées par la suite. En particulier, nous insisterons sur quelques aspects de théorie de l'information. Cette théorie a été fondée par Claude Shannon peu après la seconde guerre mondiale. Elle a été présentée dans un premier temps sous le nom de théorie mathématique des communications. L'objectif de l'auteur était de donner un cadre théorique rigoureux aux problèmes liés à la représentation et à la transmission de l'information. Ceci l'a en particulier amené à définir les notions *d'entropie* et de *capacité*. Il présente alors un schéma de transmission de la forme présentée sur la figure 1.1. Ce schéma définit un système de transmission élémentaire, où une source à transmettre est dans un premier temps compressée et dans un second temps protégée par un code de canal. Ce schéma est motivé par l'énoncé du théorème de séparation. Celui-ci stipule, sous certaines hypothèses, que l'opération consistant à compresser une source puis ensuite à la protéger par un code de canal est optimale. L'optimalité est à comprendre par rapport aux limites de compressibilité et capacité, respectivement, de la source et du canal. Ces limites théoriques ne peuvent être dépassées.

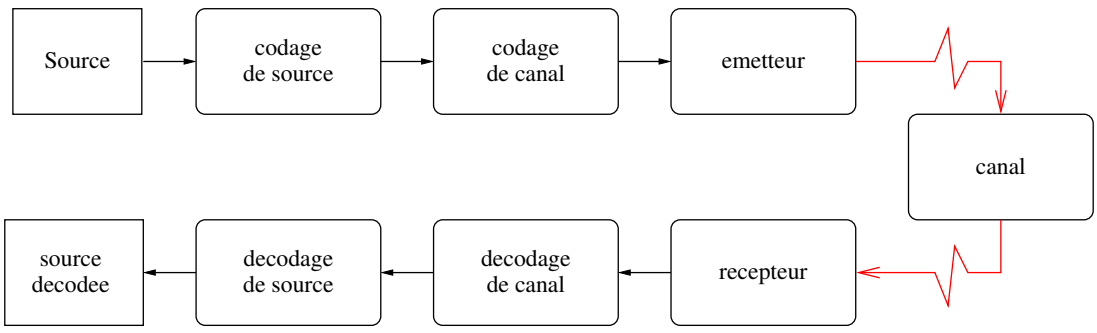


FIG. 1.1 – Schéma de transmission séparé.

Notons d'abord que l'intérêt d'un schéma séparé est d'offrir une grande flexibilité dans la conception des systèmes de communication. En effet, dans ce type de schéma, l'opération de codage de source est uniquement fonction de la source de données et ne nécessite pas la connaissance des propriétés du canal, qui par ailleurs peut ne pas être déterminé au moment du codage de source. De la même manière, le codage de canal est optimisé pour le canal considéré, sans que le codage de source n'ait à être modifié ni même que la manière dont il a été réalisé n'ait besoin d'être connue.

Nous reviendrons sur les limitations des hypothèses de ce théorème dans le chapitre 2. Précisons juste que le schéma séparé a motivé la catégorisation des problèmes rencontrés en théorie de l'information en deux types :

1. les problèmes liés au codage de source,
2. les problèmes liés au codage de canal.

Cette représentation a amplement conditionné la manière dont les systèmes de transmission et les protocoles associés ont été définis. Aujourd'hui, la dualité forte entre les problèmes de codage de source et les problèmes de codage de canal rendent quelque peu caduque cette partition. Cependant on ne peut nier son caractère pédagogique.

Ce chapitre est avant tout une entrée en matière pour le lecteur. Il effectue quelques rappels liés à la théorie de l'information, mais il n'a pas la prétention d'être un cours sur le sujet. Le livre de référence dans le domaine de la théorie de l'information est l'ouvrage de Joy A. Thomas et Thomas Cover [CT91]. Il existe également des livres disponibles en ligne diffusés par des auteurs de renom. Citons en particulier les ouvrages de David MacKay [Mac03] et de Robert M. Gray [Gra90]. Quelques rappels sur les chaînes de Markov sont ensuite proposés dans la partie 1.2. Les codes de source les plus "classiques" sont ensuite introduits. Enfin, et parce que l'objet de cette thèse est la transmission et le décodage de codes sources en présence d'erreurs, nous introduisons la notion de capacité de canal et les différents modèles de canaux qui seront utilisés dans les différentes contributions.

1.1 Entropie et Information mutuelle

Dans cette section, un certain nombre de définitions liées à des quantités couramment rencontrées en théorie de l'information seront proposées. Plutôt que d'énoncer la liste des nombreuses propriétés associées à ces quantités, nous essaierons de fournir une représentation intuitive de ces notions. Les preuves et développements de ces propriétés sont fournies, par exemple, dans [CT91].

Soit X une variable aléatoire à valeurs dans un alphabet $\mathcal{A}$. On note $\mathbb{P}(X = x)$ la probabilité de l'événement $X = x$.

Définition 1.1 *On appelle* innovation *de l'événement $X = x$ la quantité $\mathbb{I}(X = x)$ définie telle que*

$$\mathbb{I}(X = x) = -\log_2(\mathbb{P}(X = x)). \tag{1.1}$$

On aurait également pu qualifier ce terme de *quantité d'information* apportée par l'événement $X = x$. Cette quantité est positive ou nulle, ce qui correspond à l'intuition qu'une réalisation diminue l'incertitude. La nullité de cette valeur caractérise un événement qui arrive de manière certaine, i.e. avec une probabilité égale à 1. De la même manière, une valeur de l'innovation qui sera infiniment grande caractérise un événement impossible. Le choix de la base 2 pour le logarithme est arbitraire mais commode. En effet, ce choix implique que l'innovation correspondant à la réalisation d'une variable binaire de loi uniforme est égale à 1. Cette quantité intuitive peut alors être vue comme l'unité élémentaire d'innovation. Le coût requis de représentation d'une telle variable s'appelle le *bit* (binary unit).

Propriété 1.1 *Pour une loi uniforme sur un alphabet de cardinal $|\mathcal{A}| = 2^n$, nous avons*

$$\mathbb{I}(X = x) = n \text{ (bits).} \tag{1.2}$$

Définition 1.2 *L'entropie $H(X)$ d'une variable aléatoire X est l'espérance de l'innovation $\mathbb{E}(\mathbb{I}(X = x))$ de la variable aléatoire X et est donnée par*

$$H(X) = \sum_{a \in \mathcal{A}} \mathbb{P}(X = x)\, \mathbb{I}(X = x). \tag{1.3}$$

L'entropie peut intuitivement se comprendre comme la quantité de hasard associée à la variable aléatoire considérée. Elle correspond également à un coût de codage moyen minimal pour représenter une suite de réalisations d'une variable aléatoire. Cette quantité est toujours positive et est nulle pour un événement certain.

Lorsque deux événements X et Y ne sont pas indépendants, on peut mesurer l'espérance de l'innovation du couple (X, Y). La quantité associée est appelée *entropie jointe* et est notée $H(X, Y)$. Cette notion s'étend bien entendu à des n-uplets tels que $n \geq 2$. On peut également calculer l'innovation $\mathbb{I}(X|Y)$ associée à l'événement conditionnel $X|Y$. L'espérance de cette innovation est appelée entropie conditionnelle et est définie par

$$H(X|Y) = \frac{H(X, Y)}{H(X)}. \tag{1.4}$$

L'entropie conditionnelle peut également se voir comme l'innovation moyenne de l'événement X sachant que l'événement Y est connu. Il est alors intéressant de considérer la quantité moyenne d'innovation partagée par deux événements. Cette quantité est appelée *information mutuelle* et est définie par

$$I(X, Y) = H(X) - H(X|Y). \tag{1.5}$$

L'information mutuelle peut également se voir comme l'impact de la connaissance de Y sur l'innovation moyenne associée à la variable aléatoire X. Bien entendu, si les événements X et Y sont indépendants, l'information mutuelle est nulle. À l'opposé, si X et Y sont le même événement, l'information mutuelle $I(X,Y)$ est égale à l'entropie de la source. C'est pourquoi l'entropie est également parfois appelée information propre (*self-information*).

Dans un schéma de codage classique, on cherche généralement à transmettre une séquence de variables aléatoires. Une telle séquence est appelée une *source*. Nous considérons essentiellement des sources dont la loi, conditionnelle ou pas, est invariante dans le temps. On dit d'une telle source qu'elle est *stationnaire*.

À des fins d'analyse, les signaux aléatoires que nous manipulons doivent être modélisés. Précisons d'emblée que les techniques de codage de source qui s'appliquent à des sources quelconques sans modèle *a priori* sont qualifiées *d'universelles* (universal source coding).

Dans cette thèse, toutes les sources considérées sont des sources dont le modèle est supposé parfaitement connu à l'encodage. Les sources sont alors modélisées comme des processus qui suivent des lois particulières paramétrables. En particulier, l'hypothèse d'un modèle bayésien est souvent retenue pour modéliser les processus. Nous adoptons également ce choix, qui est largement motivé dans [Rob01].

1.2 Chaînes de Markov et chaînes de Markov cachées

Un certain nombre de thèmes que nous allons considérer font apparaître des chaînes de Markov et des chaînes de Markov cachées[1]. Rappelons qu'un processus stochastique est une suite de variables aléatoires indexées par une variable à temps discret, notée t. En particulier, nous supposerons que $t \in \mathbb{N}^*$. Dans la suite, nous allons essentiellement considérer des processus markoviens, définis par la propriété ci-dessous.

Définition 1.3 *Un processus de Markov d'ordre n est un processus tel que*

$$\mathbb{P}(S_t|S_{t-1},\ldots,S_2,S_1) = \mathbb{P}(S_t|S_{t-1},S_{t-2},\ldots,S_{t-n}). \tag{1.6}$$

Une chaîne de Markov est un processus de Markov d'ordre 1 à valeur dans un alphabet discret, en particulier fini. Lorsque l'espace des valeurs est fini, la chaîne de Markov est dite finie. Ainsi, pour une chaîne de Markov, la loi de probabilité de la variable aléatoire S_t à un instant t ne dépend que de la réalisation de S_{t-1} à l'instant

[1] D'une manière plus générale, ils peuvent se formaliser avec un formalisme markovien. Un réseau bayésien est un champ de variables aléatoires liées entre elles sous des hypothèses de relations d'indépendance conditionnelle. Le lecteur pourra trouver une introduction didactique aux réseaux bayésiens dans un contexte de codage dans la thèse d'Arnaud Guyader [Guy02].

précédent. Cette réalisation suffit à capter toute la mémoire liée aux réalisations passées, ce qui justifie la représentation des dépendances sous la forme proposée dans la figure 1.2.

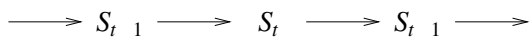


FIG. 1.2 – Chaîne de Markov

L'ensemble des probabilités $\mathbb{P}(S_t|S_{t-1})$ définit alors une matrice carrée M_t appelée matrice de transition de probabilités ou *noyau de transition*. Si de plus cette matrice ne dépend pas de l'instant t considéré, alors cette chaîne est dite *homogène* ou *invariante*. Une chaîne de Markov homogène est entièrement définie par son noyau de transition M et un état initial. Cette représentation matricielle est pratique. En effet, si les probabilités des états à un instant t sont représentées par un vecteur π_t, alors le vecteur de probabilités π_{t+1} à l'instant $t+1$ se calcule comme le produit $M \times \pi_t$ de la matrice correspondant au noyau de transition et du vecteur π_t.

Note : on suppose ici que les colonnes de la matrice M sont elles-mêmes des lois de probabilité discrètes. En d'autres termes, tous les éléments sont positifs et leur somme est égale à 1. L'élément $M(j,i)$ indexé par la ligne j et la colonne i représente alors la probabilité $\mathbb{P}(S_t = a_i|S_{t-1} = a_j)$. Cette convention matricielle permet alors d'écrire $\mathbb{P}(S_{t+k} = a_j|S_t = a_i) = M^k(j,i)$.

Un processus de Markov d'ordre n peut être ramené à un processus de Markov d'ordre 1 par vectorisation de la source. En effet, si $(X_t)_t$ est un processus de Markov d'ordre n, on peut définir la variable aléatoire

$$Z_t = (X_t, X_{t-1}, \ldots, X_{t-n+1}) \tag{1.7}$$

et remarquer que

$$\begin{aligned}
\mathbb{P}(Z_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{t-n}) &= \mathbb{P}(X_t, X_{t-1}, \ldots, X_{t-n+1}|X_{t-1}, \ldots, X_2, X_1) && (1.8)\\
&= \mathbb{P}(X_t, X_{t-1}, \ldots, X_{t-n+1}|X_{t-1}, \ldots, X_{t-n+1}, X_{t-n}) && (1.9)\\
&= \mathbb{P}(Z_t|Z_{t-1}), && (1.10)
\end{aligned}$$

c'est-à-dire $(Z_t)_t$ est un processus de Markov d'ordre 1. En particulier, si l'espace d'état de X_t est fini, le processus Z_t prend également ses valeurs dans un espace d'états fini et est, par conséquent, une chaîne de Markov.

Lorsqu'on considère une chaîne de Markov homogène, deux propriétés sont particulièrement intéressantes : *l'irréductibilité* et *l'apériodicité*. Dans le cadre de ce document, nous nous contenterons de caractériser ces propriétés de la manière suivante

1. Une chaîne de Markov homogène est irréductible si et seulement si (ssi), pour tout couple d'état (x,y), il existe une suite finie d'états $x_0 = x, x_1, \ldots, x_i, \ldots, x_k =$

y telle que $\mathbb{P}(S_t = x_i | S_{t-1} = x_{i-1}) > 0$. Ainsi, pour une chaîne finie, on peut caractériser l'irréductibilité d'une chaîne de Markov homogène par l'existence d'un entier k tel que $M^k(j,i) > 0$. Il est également possible de garantir l'irréductibilité d'une chaîne de Markov à partir des vecteurs propres de la matrice M associés à la valeur propre 1. En effet, si cette valeur propre est associée à un unique vecteur propre positif, alors le processus correspondant est irréductible.

2. Une chaîne de Markov homogène finie est apériodique ssi les valeurs propres de module 1 associées à sa matrice de transition M sont réelles. Dans le cas contraire, elle est périodique et l'argument de la valeur propre permet de trouver la période.

Lorsque ces deux propriétés sont réunies et que la chaîne de Markov est finie[2], celle-ci définit un processus *ergodique*, c'est-à-dire un processus tel que la probabilité d'aller à un état à un autre en un nombre quelconque d'étapes est non nulle. Dans ce cas, il y a une seule solution π^* à l'équation

$$\pi = M \times \pi. \tag{1.11}$$

Cette solution est la mesure invariante (ou stationnaire) pour le noyau de transition considéré. Par définition, elle correspond à l'unique vecteur propre positif de norme 1 associé à la valeur propre 1. L'unicité de ce vecteur propre est garanti par l'ergodicité du processus. Cette loi correspond au comportement à long terme de la chaîne de Markov. On a en effet

$$\lim_{k \to \infty} M^k \to [\pi^* | \ldots | \pi^*], \tag{1.12}$$

où $[\pi^* | \ldots | \pi^*]$ est la matrice carrée dont les vecteurs colonnes sont la mesure invariante pour le processus considéré. Avec le théorème de Cesaro, on a également

$$\lim_{k \to \infty} \frac{1}{N} \sum_{k'=0}^{k-1} M^{k'} \to [\pi^* | \ldots | \pi^*]. \tag{1.13}$$

Dans les problèmes que nous allons considérer, la connaissance des états d'une chaîne de Markov n'est en général pas parfaite mais bruitée par un bruit modélisé sous la forme d'une variable aléatoire ou d'un processus. Nous n'avons alors que des *observations* des différentes réalisations d'une chaîne, comme indiqué sur la figure 1.3. Les chaînes de Markov cachées apparaissent être une sous-classe particulière des réseaux bayésiens.

Dans [EM02], les auteurs proposent un tour d'horizon des chaînes de Markov cachées, vues sous l'angle de la théorie de l'information. Outre de nombreuses références

[2] Dans le cas non fini, il faut rajouter une propriété supplémentaire.

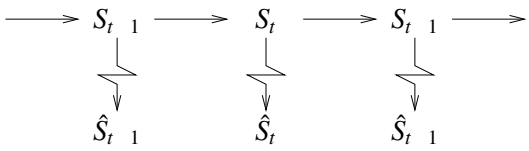


FIG. 1.3 – Chaîne de Markov cachée

bibliographiques et une présentation historique de ces processus, ils résument les différents types de problèmes considérés et les algorithmes de résolution correspondants. Dans ce mémoire de thèse, les chaînes de Markov cachées se retrouvent en particulier

- *en codage de source :* elles servent principalement de modèle à la source (par exemple, dans le chapitre 3) ;
- *en codage de canal :* la plupart du temps, l'entrée d'un code de canal est modélisée par une loi sans mémoire uniforme. C'est le processus d'encodage qui introduit artificiellement une dépendance temporelle. Par exemple, la sortie d'un codeur convolutif de canal se modélise sous la forme d'une chaîne de Markov. Le décodeur n'observe pas directement les réalisations de la chaîne de Markov mais doit les estimer à partir d'observations bruités. Le décodeur doit donc résoudre un problème impliquant une chaîne de Markov cachée.

Le problème le plus courant que nous allons rencontrer est celui consistant à estimer la séquence émise à partir d'un ensemble d'observations bruitées. L'estimateur optimal dépend de la mesure de performance. En particulier, les trois estimateurs suivants vont être considérés.

1. *Maximum a posteriori* (MAP) : cet estimateur renvoie la séquence qui a la plus grande probabilité d'avoir été émise sachant les observations $\hat{S}_1, \ldots, \hat{S}_k$:

$$\tilde{S} = \arg \max_{(S_1,\ldots,S_k)} \mathbb{P}(S_1, \ldots, S_k | \hat{S}_1, \ldots, \hat{S}_k). \tag{1.14}$$

   Cet estimateur maximise la probabilité que la séquence choisie soit correcte. Il est usuellement implémenté en utilisant l'algorithme de Viterbi [Vit67]. Précisons que ce dernier algorithme peut être vu comme un cas particulier de l'algorithme de Dijkstra [Dij59]. En effet ce dernier algorithme traite de la recherche d'un plus court chemin dans un graphe dont les étiquettes sont de valuations positives ou nulles.

2. *Maximum of posterior marginals* (MPM) : cet estimateur renvoie la séquence composée, pour chaque instant t, du symbole $\tilde{S}_t$ qui a plus plus grande probabilité d'avoir été émis, i.e.

$$\forall t,\ \tilde{S}_t = \arg \max_{\mathcal{A}} \mathbb{P}(S_t | \hat{S}_1, \ldots, \hat{S}_k). \tag{1.15}$$

Cet estimateur maximise le nombre de symboles correctement décodés. Il minimise donc le taux d'erreur symbole de la séquence. L'algorithme d'estimation proposé dans [BCJR74] permet de le calculer de manière optimale, mais précisons qu'il en existe de nombreuses approximations. Précisons également que cet algorithme se rencontre sous d'autres noms, selon la communauté scientifique considérée : algorithme forward/backward, algorithme somme/produit.

3. *Minimum of mean square error* (MMSE) : cet estimateur renvoie, à chaque instant, l'espérance du symbole, conditionnellement aux observations :

$$\forall t,\ \tilde{S}_t = \mathbb{E}(\mathbb{P}(S_t|\hat{S}_1, \ldots, \hat{S}_k)). \tag{1.16}$$

Cet estimateur minimise la reconstruction au sens de la norme euclidienne. Il suppose donc qu'une espérance puisse être calculée sur l'alphabet $\mathcal{A}$ et que la valeur reconstruite ait un sens. En effet, elle n'appartient pas forcément à l'ensemble de départ.

1.3 Codage de source

Nous nous intéressons maintenant aux techniques de représentation de l'information, appelées techniques de *codage de source*. Soit $\mathbf{S}$ une source qui prend ses valeurs dans un alphabet $\mathcal{A}$. L'objet du codage de source est de décrire $\mathbf{S}$ de la manière la plus compacte possible en utilisant des symboles à valeurs dans un alphabet $\mathcal{A}'$. La plupart du temps, $\mathcal{A}'$ est, pour des raisons pratiques, l'alphabet binaire $\{0, 1\}$. Soit $\mathcal{A}'^k$ l'alphabet produit, i.e. l'espace des séquences de longueur k à valeur dans $\mathcal{A}'$. Dans la suite, la longueur d'une séquence $\mathbf{s}$ est notée $L(\mathbf{s})$. L'ensemble $\mathcal{A}'^0$ contient conventionnellement un seul élément, la séquence vide notée ε. L'ensemble des séquences de longueur quelconque de $\mathcal{A}'$ se note souvent

$$\mathcal{A}'^* \triangleq \bigcup_{k=0}^{\infty} \mathcal{A}'^k. \tag{1.17}$$

Un code de source peut alors se représenter comme une fonction[3] qui associe à un symbole d'un alphabet source $\mathcal{A}$ une valeur dans $(\mathcal{A}')^*$:

$$\begin{array}{ccc} \mathcal{A} & \rightarrow & \mathcal{A}'^* \\ a & \mapsto & \mathcal{C}(a). \end{array} \tag{1.18}$$

Ce type de code associe à un élément de l'espace, i.e. un mot de $\mathcal{A}$ de longueur 1, un mot de l'espace d'arrivée. Si la fonction $\mathcal{C}$ est injective, alors à tout élément de $\mathcal{A}'$ est

[3] en théorie de l'information, les codes sont souvent présentés comme des ensembles.

associé au plus un antécédent dans $\mathcal{A}$. Le codage est alors dit *réversible* ou *sans perte*. Si ce n'est pas le cas, alors le codage est dit *irréversible* ou *avec pertes*. La première étape d'un système de codage de source consiste généralement à obtenir un alphabet discret. Elle est généralement appelée *quantification* [GN98]. Nous nous intéresserons en particulier au codage de source réversible. Précisons que le premier code entropique considéré a été proposé par Shannon lui-même, et porte son nom.

La définition proposée dans l'équation 1.18 peut être étendue pour permettre d'avoir un nombre variable d'éléments dans l'alphabet d'entrée. Il suffit en effet de regarder $\mathcal{A}$ comme un alphabet produit associé à la source et d'encoder les symboles en les prenant par plusieurs. Il est également possible d'écrire ce fait de manière plus explicite,

$$\begin{array}{ccc} \overline{\mathcal{A}} \subset \mathcal{A}'^{*} & \rightarrow & \mathcal{A}'^{*} \\ a & \mapsto & \mathcal{C}(a), \end{array} \tag{1.19}$$

où a représente alors un mot de $\overline{\mathcal{A}}$, qui est un sous-ensemble de $\mathcal{A}^{*}$.

La performance en compression d'un code est alors donnée par le taux de compression, défini comme l'espérance $\mathbb{E}\left(\frac{L(S)}{L(E)}\right)$. du nombre $L(S)$ de symboles émis par le nombre $L(E)$ de symboles absorbés Afin de refléter les cardinaux différents des alphabets $\mathcal{A}$ et $\mathcal{A}'$, il est préférable de le calculer de manière à permettre une comparaison directe avec l'entropie de la source :

$$\text{rendement de compression } = \mathbb{E}\left(\frac{L(S)}{L(E)}\right) \times \log_2(|\mathcal{A}'|) \text{ bits.} \tag{1.20}$$

1.3.1 Catégorisation des codes de source

1.3.1.1 Codes à longueur fixe

Ce sont des codes tels que le nombre d'éléments en entrée et en sortie du code sont constants :

$$\exists l_{\mathcal{A}}, l_{\mathcal{A}'},\ \forall a \in \overline{\mathcal{A}}, L(a) = l_{\mathcal{A}}, L(\mathcal{C}(a)) = l_{\mathcal{A}'}. \tag{1.21}$$

On les notes CLF (ou FLC pour *fixed length codes*) ou *F-to-F*. Le plus souvent, un seul élément est pris en entrée ($l_{\mathcal{A}} = 1$). Ces codes sont utiles lorsqu'aucune hypothèse n'est connue sur la source, car elle garantit que la longueur moyenne (et maximale) des mots de codes produits sera égale exactement à la constante $l_{\mathcal{A}'}$. Ce code n'est pas, à proprement dit, un code de compression. Il est optimum uniquement pour une source uniforme sur $\mathcal{A}^{l_{\mathcal{A}'}}$ et si $|\mathcal{A}|^{l_{\mathcal{A}'}} = |\mathcal{A}'|^{l_{\mathcal{A}}}$.

1.3.1.2 Codes à longueur fixe vers longueur variable

Ces codes, notés *F-to-V*, sont appelés codes à longueur variable (CLV) par abus de langage, car ils sont très utilisés. Ce sont les codes qui prennent un nombre constant

d'éléments en entrée et qui sortent un nombre variable d'élément en sortie. Là encore le cas $l_{\mathcal{A}} = 1$ est très souvent considéré. Ce type de code inclut les codes de Huffman, les codes de Shannon et d'une certaine manière les codes arithmétiques. Il inclut également les versions généralisées de ces codes (codes de Huffman et de Shannon généralisés), c'est-à-dire les versions vectorielles des codes de Huffman et de Shannon, dont les performances en compression continuent d'être étudiées [YY02].

Les codes tels $l_{\mathcal{A}} > 1$ sont souvent appelés *Bloc vers Variable*, afin de les différencier du cas $l_{\mathcal{A}} = 1$. Les versions généralisés des codes de Huffman et de Shannon entrent bien entendu dans cette catégorie.

1.3.1.3 Codes à longueur variable vers longueur fixe

Ces codes prennent un nombre variable d'éléments en entrée et génère un nombre constant de symboles en sortie. Ils sont notés en abrégé *V-to-F*. Un exemple de codes rentrant dans cette catégorie sont les codes de Tunstall [Tun67] et les codes de Tunstall généralisés. La propriété de convergence de ces derniers lorsque la longueur de la séquence tend vers l'infini a été étudié en particulier dans [SG97].

1.3.1.4 Codes à longueur variable vers longueur variable

Cette catégorie de codes, notée *V-to-V*, est la plus générale et inclut tous les types de codes précédents. La propriété de convergence de certains codes rentrant dans cette catégorie de manière stricte, à savoir les codes de Tunstall-Huffman et les codes de Tunstall-Shannon-Fano, est étudié dans [SS02]. Il est ainsi démontré que ces codes permettent de converger plus rapidement vers l'entropie que des codes de Huffman lorsque la longueur de la séquence tend vers l'infini. Des limites supérieures et inférieures de redondance qui ne sont pas liées à des techniques de construction particulières ont été proposées par Khodak [Kho72], sans toutefois que des techniques de construction atteignant ces bornes soient explicitées. Même si la recherche de techniques de construction de codes de longueur variable vers variable reste un domaine actif (e.g., [DS04]), ces codes sont peu utilisés en pratique dans leur version la plus générale car ils requièrent *a priori* des structures de données complexes.

1.3.2 Codes de source usuels

Dans cette section, nous allons essentiellement considérer des codes à longueur variable. Il est intéressant de considérer les codes de ce type qui permettent d'optimiser des critères de performances répondant à un besoin particulier. Les critères de performance que nous considérerons plus particulièrement sont les suivants :

1. espérance de la longueur minimale,
2. espérance de la longueur minimale sous contrainte d'une longueur maximale,

3. espérance de la longueur minimale sous contrainte d'ordre lexicographique.

Pour des codes à longueur variable prenant en entrée un mot de longueur 1, la performance en compression d'un code à longueur variable est donnée par l'espérance $\mathbb{E}(L(\mathbf{S}))$ de la longueur $L(\mathbf{S})$ de représentation du mot de code de $\mathcal{A}'$, exprimée en nombre de symboles de l'alphabet $\mathcal{A}'$ ou en un nombre de bits équivalent. Le code de longueur variable optimum $\mathcal{C}^*$ pour le critère de la longueur minimale est donc donné par

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \mathbb{E}(L(\mathbf{S})). \tag{1.22}$$

Il apparaît clairement que l'espérance dépend de la distribution de probabilité sur l'espace des séquences. Pour résoudre le problème de manière optimale, un pré-requis est d'avoir la connaissance parfaite de la distribution de probabilité $\mathbb{P}(\mathbf{S})$. Cette hypothèse est très souvent retenue, et d'ailleurs l'entropie d'une variable aléatoire est calculée relativement à une connaissance de ces probabilités. Les codes qui ne font pas l'hypothèse d'un modèle de source connu *a priori* sont appelés codes *universels* (Universal source codes) et les plus populaires sont probablement les différents codes de Lempel-Ziv [ZL77][ZL78] ou plus récemment, les Grammar Codes [KY00]. Cette problématique ne sera pas abordée parce qu'elle sort du cadre de cette thèse. Précisons néanmoins que la notion d'information proposée par Claude Shannon n'est pas toujours suffisante puisqu'elle nécessite la connaissance parfaite d'un modèle de probabilité donné. C'est pourquoi Kolmogorov a définit la complexité d'une séquence comme étant la taille du plus petit programme décrivant la séquence (voir le chapitre 7 de [CT91] pour une introduction succincte). Cette taille est exprimée pour une machine de Turing mais s'applique, à une constante près, à toute machine à états finis.

1.3.2.1 Codes optimaux et limitations pratiques

Sous l'hypothèse d'un modèle de source donnant la probabilité de toute séquence, le code optimal $\mathcal{C}^*$ pour le critère de l'espérance de la longueur de l'équation 1.22 est le code de Huffman [Huf52]. Ce code est construit par une approche ascendante, contrairement au codes initialement proposés par Shannon. Les codes optimaux ne sont pas uniques. D'une part des étiquettes (0 ou 1) associées aux transitions de l'arbre de codage peuvent être inversées. Dans ce cas les longueurs associées aux symboles aux différents symboles restent inchangées. Mais il est également possible d'obtenir des codes de longueur moyenne égale avec des longueurs associées aux symboles qui sont différentes. Dans ce cas l'arbre de hauteur minimale est généralement privilégié.

Les codes de Huffman sont très largement utilisés en pratique pour des alphabets de petite taille, car les encodeurs et décodeurs qui leur sont associés sont très efficaces. Leur version généralisée demande une complexité qui est importante [Tja00] et il n'est pas possible de les utiliser sur des séquences de grande taille, car il n'existe

pas de description implicite de ces codes. Il est donc nécessaire de stocker ou de calculer l'arbre de codage, et ce pour toutes les longueurs de séquence possibles. Ce n'est pas réalisable en pratique car le nombre d'éléments d'un alphabet produit croît de manière géométrique avec la longueur de la séquence. Par ailleurs, la complexité associée à la construction d'un code de Huffman généralisé est en $\mathcal{O}(N_f \log N_f)$ où N_f représente le nombre de feuilles de l'arbre correspondant. Le coût de calcul est donc également prohibitif. Les codes de Huffman sont donc principalement utilisés sur l'alphabet de base de la source et non sur un alphabet de vecteurs d'éléments de la source. Les performances obtenues demeurent très correctes lorsque la taille de l'alphabet est importante [BK93].

Pour le critère de l'équation 1.22, les techniques de compression se comparent à la performance limite, donnée par l'entropie de la source. Les codes de Huffman comme les codes de Shannon, appliqués à des vecteurs de symboles, atteignent asymptotiquement l'entropie de la source lorsque la longueur de la séquence tend vers l'infini. Ce ne sont pas les seuls. En effet, les codes de Shannon généralisés ont également de très bonnes performances asymptotiques [Szp00]. Cependant, comme nous l'avons déjà souligné, il n'est pas possible d'utiliser cette propriété en pratique car les codes sont trop gros et trop complexes à générer.

1.3.2.2 Codage arithmétique

Le succès des codes arithmétiques [Pas76][Ris76] s'explique en partie par leur capacité à se rapprocher de l'entropie avec une complexité linéaire. L'idée directrice de ces codes est assez ancienne [Sha48][Abr63]. Ces codes sont des codes bloc vers variable qui sont asymptotiquement optimaux lorsque la taille du bloc tend vers l'infini. Leur optimalité suppose que la précision machine soit infinie. En pratique, le coût algorithmique des multiplications et divisions en précision arbitraire rendent les algorithmes d'encodage et de décodage trop complexes. Néanmoins des versions en précision finie du codage arithmétique sont très utilisées en pratique, car ce codage présente les avantages majeurs suivants :

- ils permettent d'approcher l'entropie de la source avec des complexités d'encodage et de décodage qui sont linéaires par rapport à la longueur de la séquence ;
- ils sont très flexibles dans le sens où la probabilité peut être fournie pour chaque symbole, ce qui permet d'introduire des modèles adaptatifs ;
- en précision finie, ils permettent un décodage quasi-instantané, c'est-à-dire que le nombre de bits à lire avant décodage d'un symbole est bornée par une constante de taille raisonnable.

1.3.2.3 Codes lexicographiques

Soit un ensemble $\mathcal{X}$ muni d'un ordre $\prec$. L'ordre lexicographique dérivé de l'ordre $\prec$ (ou plus simplement, lorsque cela ne présente aucune ambiguïté, l'ordre lexicographique) est également noté $\prec$. Cette relation d'ordre est définie sur l'ensemble des mots de $\mathcal{X}^*$ et est définie de la manière suivante : soit $\mathbf{x} = x_1, \ldots, x_{L(x)}, \mathbf{y} = y_1, \ldots, y_{L(y)} \in \mathcal{X}^*$,

$$\mathbf{x} \prec \mathbf{y} \text{ ssi } \exists n \in \mathbb{N} \,/\, \begin{cases} \forall n' < n,\ x_{n'} = y_{n'} \\ x_n < y_n. \end{cases} \tag{1.23}$$

Nous allons considérer les codes qui préservent un ordre dans le domaine transformé. Soit $\prec_{\mathcal{A}}$ un ordre total sur $\mathcal{A}$ lexicographiquement étendu à $\mathcal{A}^*$. On considère de même un ordre $\prec_{\mathcal{A}'}$ sur $\mathcal{A}'$ et l'ordre lexicographique associé sur $\mathcal{A}'^*$. Un code est dit lexicographique si et seulement si

$$\forall x, y \in \mathcal{A}^*,\ x \prec y \Rightarrow \mathcal{C}(x) \prec \mathcal{C}(y). \tag{1.24}$$

Les codes lexicographiques sont intéressants dans la mesure où ils permettent de comparer deux mots de $\mathcal{A}^*$ directement dans le domaine transformé. Ceci est en particulier utile lorsqu'on veut effectuer une requête comparative dans un ensemble trié, dans une application de base de données. Il est en effet possible d'effectuer la recherche dichotomique d'une chaîne en la compressant et en la comparant à des données compressées, plutôt que de décompresser toutes les chaînes auxquelles on veut se comparer lors de la recherche. Comparer dans le domaine compressé permet en outre de diminuer le temps de comparaison. En effet l'espérance du temps de comparaison est d'autant plus faible que l'espérance de la longueur des chaîne est courte. En première approximation le facteur constant de réduction du temps de requête correspond au taux de compression.

Le code optimum pour le critère de l'équation 1.22 sous la contrainte supplémentaire que le code soit lexicographique est le code de Hu-Tucker [HT71]. Ces codes seront utilisés dans le chapitre 6.

1.3.3 Un autre type de code source

Il existe de nombreuses autres techniques de codage de source. Citons parmi celles-ci le run-length encoding (RLE) [Gol66]. C'est un algorithme qui est adapté à la présence de longues séquences d'un symbole identique. C'est souvent le cas pour le 0, par exemple. L'idée est très proche de celle de la factorisation du + en $\times$. En effet, les symboles sont encodés en couples de la forme (nombre d'occurrence, symbole). Ainsi, la séquence 0000112000 est encodée en $(4,0)(2,1)(1,2)(3,0)$. Cet algorithme, en dépit

de sa simplicité, est encore largement utilisé actuellement sous différentes formes dans les standards, car il est adapté à de nombreuses situations et a un coût algorithmique faible. En particulier, il est utilisé dans une version modifiée comme un pré-traitement permettant de réduire les séquences de 0.

1.4 Capacité de canal et modèles de canaux

Dans cette partie, les enjeux du codage de canal sont succinctement présentés de manière informelle. Nous introduisons ensuite la notion de capacité et les modèles de canaux que nous utiliserons dans la suite de ce document.

1.4.1 Enjeux du codage de canal

Le codage de canal a pour objet de protéger les données contre les erreurs qui peuvent survenir sur le canal physique lors de la transmission d'informations. Un moyen de réduire le nombre d'erreurs sur le canal consiste à augmenter physiquement le rapport signal à bruit sur le canal. Cependant il aboutit à une augmentation de l'effet Joule, ce qui limite son usage. Le critère de quantité d'énergie dépensée par bit d'information utile a une signification pratique et est très souvent pris comme référence. C'est pourquoi les performances respectives des codes de canal sont comparées en terme de quantité d'énergie requise pour transmettre un bit d'information avec une probabilité d'erreur asymptotiquement nulle.

Ainsi, le codage de canal a souvent pour objectif de minimiser la quantité d'énergie dépensée par bit utile sous la contrainte de ne pas dépasser un certain taux d'erreur. La notion de capacité de Shannon peut également s'exprimer en fonction de cette quantité d'énergie à apporter par bit utile et est alors notée E_b/N_0. Cette quantité est dans ce cas exprimée en décibels. Une définition plus précise et générale de la capacité sera donnée dans la section 1.4.3.

Les techniques de codage de canal permettent de réduire cette quantité pour atteindre un taux d'erreur résiduel cible. Elles se basent sur des structures algébriques qui permettent la détection et/ou la correction d'erreurs. Plutôt que d'entrer dans une classification de toute façon non exhaustive des techniques de codage de canal, nous invitons le lecteur à consulter le livre de Lin et Costello [LC04], recommandé dans [Bla05]. Les enjeux industriels autour des techniques de codage de canal sont importants, dans la mesure où chaque gain en terme de décibels sur le canal, même faible, engendre des économies importantes.

Notons cependant que si les techniques de codage de canal permettent effectivement de réduire la quantité d'énergie par bit utile, elles induisent une diminution de la bande passante qui n'apparaît pas dans l'indicateur de performance E_b/N_0. Par ailleurs, elles nécessitent d'effectuer des traitements plus complexes au niveau du décodeur. Ce problème de complexité est omniprésent en codage de canal. En effet, de-

puis Shannon il existe des procédés constructifs permettant d'atteindre la limite théorique de capacité de canal. Or, les procédés en question ne peuvent pas s'appliquer en raison de la trop grande complexité algorithmique qui leur est associée. Ainsi la recherche en codage de canal porte son attention sur la quête de procédés d'encodage approchant les performances de la limite théorique avec une complexité faible. C'est pourquoi les turbo-codes [BGP93] ont eu un impact rapide, car ils ont marqué l'aboutissement de cette quête. Ces codes ont d'abord été construits à partir de codes convolutifs récursifs systématiques, mais la technique turbo se généralise à d'autres codes en bloc [Pyn98]. Nous verrons également qu'elle a été considérée dans un contexte de décodage conjoint source/canal. Précisons que d'autres types de codes permettent également d'obtenir des performances proches de la limite de Shannon. En particulier, les codes LDPC [Gal62] introduits en 1962 par Gallager sont également très populaires.

1.4.2 Formulation théorique et capacité

Un canal est un objet mathématique qui représente formellement le bruit que subit un signal lorsqu'il est transmis par un procédé réel. Il peut être vu comme une fonction qui associe à tout vecteur d'entrée $\mathbf{X} \in \mathcal{X}$ un ensemble un vecteur de sortie $\mathbf{Y} \in \mathcal{Y}$ avec une probabilité $\mathbb{P}(\mathbf{Y}|\mathbf{X})$ donnée, comme indiqué sur la figure ci-dessous.

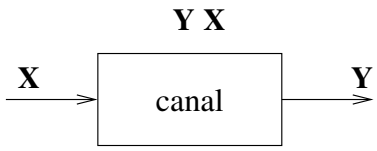


FIG. 1.4 – Canal

Les ensembles $\mathcal{X}$ et $\mathcal{Y}$ dans lesquels $\mathbf{X}$ et $\mathbf{Y}$ peuvent être finis ou continus. Ainsi $\mathbb{P}(\mathbf{Y}|\mathbf{X})$ représente soit une loi discrète soit une densité de probabilité. Dans ce document, les types de canaux suivants seront utilisés :

- le canal binaire symétrique,
- le canal discret sans mémoire,
- le canal gaussien.

Le canal binaire symétrique est un modèle très simple. Il est utilisé en pratique car il peut être vu comme un canal gaussien avec une décision dite *dure* (ou *hard*) sur le signe. Sa simplicité permet également de démontrer des résultats analytiques difficiles voire impossibles à obtenir avec des modèles plus complexes. Ce canal est un cas particulier des canaux discrets sans mémoire. Enfin, le canal gaussien est un modèle de canal continu très utilisé. Sa popularité est liée au théorème central limite. En effet, il est naturel de modéliser la somme de nombreux petits bruits aléatoires

suffisamment indépendants comme une loi gaussienne. Ceci est d'autant plus vrai qu'il existe généralement en pratique un traitement appelé *égalisation* -de canal- dont le rôle est, étant donné des caractéristiques données pour le canal réel (modulation, etc), de générer une séquence débiaisée de scalaires qui sont alors proches de réalisations de variables gaussiennes.

Il existe des modèles de canaux qui permettent de modéliser plus précisément les canaux réels. En particulier, l'hypothèse de stationnarité dans les modèles de canaux que nous présenterons est souvent fausse en pratique, comme pour les canaux radio-mobiles. Le canal qui correspond à Internet est également plus complexe qu'un simple canal à effacement. Trouver des modèles pertinents pour des canaux réels est un sujet de recherche en soi. C'est pourquoi nous nous contenterons de décrire ceux qui sont les plus largement répandus.

Quelque soit le canal, il est alors intéressant de connaître la quantité d'information qu'il est possible de transmettre par unité de temps. Dans [Sha48], Shannon introduit la notion de *capacité de canal* C comme la limite fondamentale qui borne la quantité d'information transmissible sur un canal avec une probabilité d'erreur arbitrairement petite. Pour un canal discret, cette quantité C est donnée par le maximum de l'information mutuelle entre $\mathbf{X}$ et $\mathbf{Y}$, pour toutes les lois possibles X en entrée, soit

$$C = \max_{p(X)} I(X, Y). \tag{1.25}$$

1.4.3 Modèles de canaux usuels et leur capacité

1.4.3.1 Canal binaire symétrique et canal à effacement

Le canal binaire symétrique (binary symmetrical channel, BSC) est représenté sur la figure 1.5. L'entrée de ce canal peut se représenter par une variable en entrée X. Ce canal accepte deux symboles en entrée ($X = 0$ et $X = 1$). Un unique paramètre p (*cross-over* probability) permet de définir parfaitement ce canal. Ce paramètre correspond à la probabilité que le symbole reçu Y ne soit pas correct.

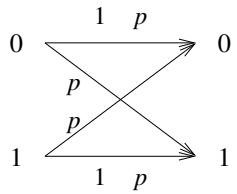


FIG. 1.5 – Canal binaire symétrique

Ainsi, ce canal peut se modéliser par la matrice de transition de probabilité, aussi

appelée *matrice de transmission*

$$\mathbb{P}(Y = y|X = x) = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}, \tag{1.26}$$

dont on déduit la capacité :

$$C = 1 - H(p). \tag{1.27}$$

Ce maximum d'information mutuelle est obtenu lorsque la loi $\mathbb{P}(X)$ de X est uniforme en entrée. Un autre canal proche du canal binaire symétrique est le canal à effacements, représenté sur la figure ci-dessous. Pour ce canal, $\mathcal{X} = \{0, 1\}$ et $\mathcal{Y} = \{0, 1, E\}$, où l'état E indique que le symbole émis est inconnu. Cet état a la probabilité p d'être reçu. Il n'existe pas d'erreur sur ce canal, ainsi si $Y = 0$ ou $Y = 1$, alors de manière déterministe $X = 0$ et $X = 1$, respectivement.

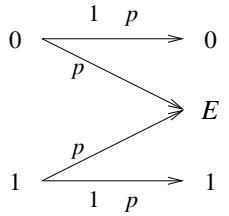


FIG. 1.6 – Canal à effacement

Ce canal est caractérisé par la matrice de transmission

$$\mathbb{P}(Y = y|X = x) = \begin{pmatrix} 1-p & 0 \\ 0 & 1-p \\ p & p \end{pmatrix}, \tag{1.28}$$

ce qui mène à

$$C = 1 - p. \tag{1.29}$$

1.4.3.2 Canal à bruit additif gaussien sans mémoire

Le canal sans mémoire à bruit additif gaussien ou, lorsque cela ne présente pas d'ambiguïté, le canal gaussien, est modélisé comme indiqué sur la figure 1.7. La source $\mathbf{X}$ peut *a priori* être continue, mais dans un système pratique il est nécessaire de la discrétiser. L'ensemble de valeur $\mathcal{X}$ est alors fini. On choisit souvent un modèle tel que $\mathcal{X} = \{-A, A\}$, qui est utile car il permet de modéliser un canal avec une modulation BPSK (*binary phase shift keying*).

Remarquons que, étant donné que le canal a une variance fixée et supposée indépendante du signal émis, on peut rendre le bruit négligeable par rapport au signal en

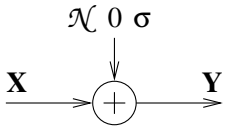


FIG. 1.7 – Canal additif gaussien sans mémoire

augmentant arbitrairement l'énergie moyenne de celui-ci. Cependant, une contrainte énergétique P est généralement fixée pour le signal d'entrée :

$$\mathbb{E}(||X||^2) \leq P. \tag{1.30}$$

Cette contrainte est largement utilisé dans les analyses théoriques [4]. Sous cette contrainte de puissance moyenne, la capacité du canal gaussien est donnée par

$$C = \frac{1}{2}\log(1 + \frac{P}{\sigma^2}). \tag{1.31}$$

Pour un signal d'entrée ne prenant que deux valeurs $+A$ et $-A$, donc pour une amplitude telle que $A = \sqrt{P}$, la probabilité d'erreur si une décision est prise sur la mesure du canal est donnée par

$$1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\sqrt{\frac{P}{\sigma^2}}} e^{-\frac{t^2}{2}} dt. \tag{1.32}$$

Ainsi, un système de transmission qui utilise le canal gaussien de cette manière, avec un signal bi-valué et en effectuant une décision en sortie sur le signe, est équivalent à un canal binaire symétrique de taux d'erreur bit donné par l'équation 1.32. Il est possible de vérifier que ce canal binaire symétrique a une capacité inférieure au canal gaussien. En d'autres termes, cette manière d'utiliser le canal gaussien est sous-optimale.

1.4.3.3 Canal discret sans mémoire

Le dernier type de canal que nous allons aborder est le canal discret. Ce canal est défini par une matrice de transmission $\mathbb{P}(\mathbf{Y}|\mathbf{X})$, pas nécessairement mais le plus souvent carrée. Hormis pour des cas particuliers, tels que le canal binaire symétrique ou le canal à effacements, il n'existe pas en général d'expression analytique connue de

[4] Dans les réseaux réels, il faudrait toutefois rajouter une contrainte de puissance maximale, liée à la capacité maximale d'échauffement des appareils de transmission.

la capacité de canal. Cependant, il est possible de la calculer numériquement au moyen de l'algorithme de Arimoto-Blahut [Ari72][Bla72], dont la vitesse de convergence est en $\mathcal{O}(\frac{1}{N})$ [Gal94] et peut être améliorée sous certaines conditions [Say00].

1.5 Conclusion

Dans cette partie, nous avons introduit les notions qui seront utilisées dans cette thèse : l'estimation bayésienne, le codage de source et quelques éléments de capacité de canal. Étant donné que cette thèse traite majoritairement du codage entropique, nous avons plus particulièrement insisté sur les notions discrètes.

Chapitre 2

État de l'art et contributions en codage conjoint source/canal

La séparation des systèmes de transmission en deux modules distincts, respectivement le codage de source et le codage de canal, est justifiée par le théorème de séparation. Cependant, les hypothèses de ce théorème ne sont pas vérifiées en pratique. Ces hypothèses sont rappelées dans la section 2.1 et les limitations qui en découlent sont soulignées. Dans la section 2.2, nous proposons de classifier les techniques de codage de source robuste et de codage conjoint source/canal en fonction du contexte dans lequel elles peuvent être appliquées. Des techniques efficaces de codage de source et de codage conjoint source/canal sont alors présentées dans la section 2.3, avant de poursuivre dans la section 2.4 par le résumé et le positionnement des techniques proposées dans cette thèse.

2.1 Principales limitations du principe de séparation

Le théorème de séparation est un résultat asymptotique qui n'est valable que pour des séquences de longueur arbitrairement grande et si la capacité de calcul disponible est non bornée. Il suppose en outre que les performances d'un système de transmission sont mesurées par la probabilité de récupérer parfaitement un signal, c'est-à-dire pour le critère du taux d'erreur séquence. Il suppose également que la communication est point-à-point. Enfin, il fait l'hypothèse qu'il est possible de modéliser parfaitement les propriétés du canal et que celles-ci ne varient pas au cours du temps. Ainsi, les limitations suivantes apparaissent en pratique.

1. Il n'est pas possible d'attendre un temps infini avant d'émettre des données. Or le théorème de séparation n'est plus valable si *une contrainte de délai* existe. De même, il n'est pas applicable si le volume de données à transmettre est borné et est faux, en particulier, si le volume de données est petit. Or, les protocoles de transmission des réseaux de communications les plus répandus (Ethernet,

ATM, SONET/SDH) manipulent des paquets de taille relativement petite. Ces petites tailles augmentent la flexibilité des algorithmes de routage et réduisent les latences, mais limitent la possibilité d'utiliser des codes correcteurs longs, les seuls qui se rapprochent raisonnablement des limites asymptotiques.

2. La mesure de performance du taux d'erreur séquence ne reflète pas toujours au mieux la mesure subjective de l'utilisateur du système de transmission en bout de chaîne. En particulier, dans le cas d'applications multimédia, ce qui importe n'est pas que le signal soit parfaitement restitué, mais plutôt qu'il ressemble à l'original. Par *ressembler*, nous entendons ici pour une mesure subjective humaine ou pour une mesure objective sensée représenter ladite mesure subjective. Une telle mesure objective a souvent des propriétés qui sont reliées à une distance, prise dans son sens mathématique. Il est courant d'utiliser pour cette dernière l'erreur quadratique moyenne ou, de manière équivalente, le rapport signal à bruit crête-à-crête (PSNR : *Peak Signal To Noise Ratio*).

3. Si la source à transmettre est à support non discret, alors représenter de manière parfaite un unique élément de cette source requiert un nombre de bits infini. Dans ce cas, et en mettant de côté les problèmes liés aux erreurs sur le canal, il est nécessaire d'optimiser la manière dont ce nombre -fini- de bits est utilisé pour représenter au mieux le signal pour un critère donné. C'est le cadre de la théorie débit-distorsion, qui inclut comme un cas particulier le codage de sources à support discret discuté dans la section 1.3. Cette théorie est introduite dans le chapitre 13 de [CT91]. Le cadre de cette théorie inclut les techniques dites de *quantification*, dont nous n'utiliserons que les plus simples dans le cadre de cette thèse.

4. Comme indiqué dans la section 1.4, les modèles de canaux standards ne reflètent pas parfaitement les canaux réels. Obtenir une telle modélisation est difficile, d'autant plus que les canaux considérés ne sont pas stationnaires, qu'il s'agisse d'Internet ou des canaux sans fil (*wireless channels*). Il est donc important de pouvoir disposer de techniques de codage/décodage qui soient robustes pour une très large classe de canaux. Bien que certains codes, tels que les codes LDPC [Gal62], répondent relativement bien à ce critère, ils ne s'appliquent qu'à des séquences très longues. Par ailleurs, si le canal n'est pas stationnaire et en l'absence d'un canal de retour, ils ne peuvent s'adapter de manière optimale au canal.

5. Le théorème de séparation démontre l'existence d'au moins un schéma de codage, mais ne garantit pas que la complexité algorithmique associée puisse être traitée par une machine de Turing. En présence d'une contrainte de complexité, exprimée par exemple en termes des nombres d'affectations, d'opérations arithmétiques et de déréférencement, le théorème de séparation n'est *a priori* plus valide.

D'un point de vue pratique, la capacité de calcul des appareils qui effectuent le codage de source et de canal peut être très différente au codage et au décodage. Par exemple, les terminaux mobiles sont généralement moins puissants que les stations de base des réseaux d'opérateurs ou des ordinateurs individuels. Dans ces conditions, la redondance du signal peut ne pas être entièrement exploitée pour améliorer les performances en compression lorsque le terminal est l'encodeur, en raison du coût algorithmique important ou de la mémoire requise pour améliorer les performances du codeur de source. Il reste donc de la redondance résiduelle au décodeur qui est *a priori* exploitable. Bien entendu, la séquence étant déjà transmise, l'amélioration ne concerne pas les performances en compression mais la pertinence de la reconstruction.

Enfin, pour conclure sur l'intérêt de faire du codage conjoint source-canal, soulignons que le théorème de séparation ne dit pas qu'il est nécessaire que le codage de source et le codage de canal *doivent* être effectués de manière séparée pour garantir l'optimalité du système global. Ainsi, un théorème de codage de source optimisé canal pour les codes joints basés treillis a été proposé [DG81]. Ce théorème motive les approches de quantification vectorielle optimisées canal, comme celles proposées dans [ZG90] et [FV91]. Nous reviendrons sur ce point dans la sous-section 2.3.4.

2.2 Codage de source robuste et codage conjoint source-canal : classification

Pour améliorer l'efficacité globale des systèmes de transmission, des techniques de codage de source robuste et de codage et décodage conjoint source/canal ont été proposées. Une classification de ces techniques a été proposée dans [SOD00]. Les auteurs distinguent ainsi quatre catégories de schéma de codage source/canal :

1. les schémas où le codage de canal et le codage de source sont totalement intégrés ;
2. les schémas où le codage de canal et le codage de source sont concaténés mais où le rendement des deux types de codes est optimisé pour améliorer les performances en termes du critère débit-distorsion ;
3. les schémas de protection inégale aux erreurs ;
4. les schémas de codage dits contraints où les codes de source sont modifiés pour prendre en compte la présence d'erreurs.

Cette classification ne soulève pas l'unanimité. Ainsi dans sa thèse [Xia03], Xiang dénonce le caractère arbitraire et non exhaustif d'une telle classification. Dans le présent document et contrairement à [SOD00], nous proposons de fournir une classification qui n'est pas fonction du type de méthode utilisé, mais du contexte dans lequel de telles techniques peuvent être utilisées.

Les techniques sont alors classifiées de la manière suivante :

Les techniques de codage de source robuste sont les techniques de codage de source qui sont optimales ou presque en compression. Dans ces techniques, nous incluons

- l'utilisation de codes à longueur variable avec de meilleures propriétés de resynchronisation en présence d'inversion de bits ;
- l'optimisation de l'étiquetage binaire (*index assignment*) ;
- les techniques de paquétisation qui visent à augmenter la robustesse[1].

Cette catégorie inclut uniquement la catégorie 4 de la classification de [SOD00].

Les techniques de codage conjoint source/canal sont les techniques qui laissent volontairement de la redondance au niveau du codage de source ou qui effectuent l'opération de codage de source de manière unifiée avec l'insertion d'une protection sous forme de redondance pour contrer les aléas du canal. Ces techniques incluent

- les techniques de protection inégale aux erreurs (UEP : *unequal error protection*) qui utilisent des codes correcteurs,
- la quantification (vectorielle) optimisée canal (COVQ : channel optimized vector quantization),
- les techniques de *description multiple*,
- l'ajout volontaire de marqueurs de synchronisation ou de symboles ou intervalles interdits dans les trains binaires de codes à longueur variable.

Les techniques de décodage conjoint source/canal permettent d'exploiter la redondance résiduelle d'un système de transmission séparé mais qui est sous-optimal. Ces techniques incluent

- le décodage souple de codes à longueur variable,
- le décodage conjoint source-canal de codes à longueur variable en série avec un code correcteur,
- l'utilisation d'informations prévues par un standard, par exemple les marqueurs de resynchronisation.

Les techniques de codage non séparé ne s'appliquent pas toutes de manière universelle. Ainsi les systèmes de codage et décodage conjoint nécessitent souvent d'avoir accès, au moins du côté du décodeur, à l'ensemble de la chaîne de décodage (source et canal). C'est rarement le cas en pratique, dans la mesure où le décodage de canal est souvent effectué par un appareil différent situé au niveau de la couche physique du réseau, alors que le décodage de source est effectué au niveau de l'application. C'est le principal obstacle à l'application des techniques de codage conjoint qui nécessitent les mesures du canal.

Méthode	Encodeur séparé	Décodeur séparé	Type de Canal et connaissance requise
choix de CLV plus robustes	Oui	Oui	canal quelconque à erreurs
Techniques de paquétisation	Oui	Oui	canal quelconque à erreurs
Protection inégale au niveau source	Oui	Oui	canal quelconque à erreurs
Décodage souple de CLV	Oui	Non	canal de caractéristiques connues au décodeur
Décodage conjoint source-canal de CLV	Oui	Non	canal de caractéristiques connues au décodeur
Exploitation d'informations d'un standard	Oui	Non	canal de caractéristiques connues au décodeur
Protection inégale – codes correcteurs	Non	Non	canal de caractéristiques connues à l'encodeur
Quantification vectorielle optimisée canal	Non	Non	canal de caractéristiques connues à l'encodeur
Description multiple	Non	Non	canal à perte de paquets
Ajout de marqueurs de synchronisation ou de symboles interdits	Non	Non	canal quelconque à erreurs

TAB. 2.1 – Classification des techniques de codage robuste et de codage/décodage source canal en fonction de la possibilité ou non de les appliquer si le codeur de canal (resp. le décodeur) est séparé du codage (resp., du décodage) de source.

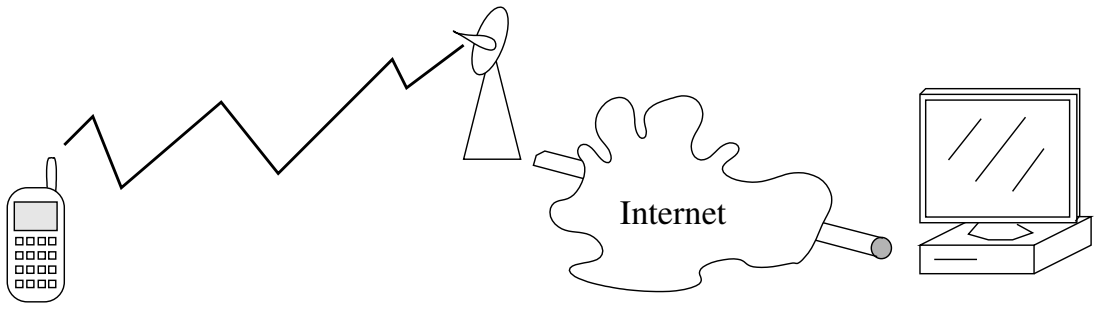


FIG. 2.1 – Communication entre un ordinateur individuel et un terminal mobile

Le tableau 2.1 résume les techniques qui s'appliquent en fonction des contraintes imposées par les applications. Les contraintes que nous considérons sont de deux types, qui correspondent respectivement à la possibilité ou non d'effectuer conjointement l'encodage ou le décodage. En effet, ceci n'est pas toujours possible. Considérons par exemple le système de communication présenté figure 2.1. Ce système de transmission entre un ordinateur personnel et un terminal mobile implique plusieurs canaux en série : Internet entre l'ordinateur et la station de base, un canal sans fil entre la station de base et le terminal mobile. La protection canal doit être adaptée à ces canaux. Or, le canal qui est ici le plus sujet aux erreurs est le canal sans fil et est donc celui qui reçoit le plus de protection. Dans le sens ordinateur → terminal mobile, l'encodage de source est effectué par l'ordinateur et ne peut donc pas être effectué de manière conjointe avec la station de base, à moins d'utiliser un *transcodage* au niveau de la station de base, c'est-à-dire de décoder et de ré-encoder le train binaire à la volée. Une telle opération est très coûteuse en ressources et est difficilement applicable en pratique. Il est néanmoins possible de décoder conjointement le train binaire au niveau du terminal mobile, car c'est le même appareil qui effectue les deux opérations de décodage. Notons que dans l'autre sens, i.e. du terminal mobile vers l'émetteur, le décodage de la partie sans fil ne peut pas être effectué conjointement sans transcodage.

Sans redondance résiduelle, il n'est pas possible de corriger des erreurs du canal. La philosophie des techniques de codage de source robuste est de limiter l'impact des erreurs pour une mesure de distorsion donnée. Ainsi, de telles techniques ne sont utiles que pour des applications où une version dégradée du signal reste utile. C'est le cas, en particulier, des signaux multimédia auxquels nous nous intéresserons par la suite, mais ce n'est bien entendu pas applicable pour la transmission de programmes exécutables. Néanmoins, lorsqu'il est possible de les utiliser, les techniques de codage de source robuste s'insèrent naturellement dans les schémas de codage séparés et

[1]Ces techniques peuvent être incluses en codage de source robuste uniquement si la paquétisation requiert un nombre de paramètres qui est faible.

peuvent être largement utilisés en pratique, car, comme indiqué dans le tableau 2.1, ces techniques ne remettent pas en cause la partie codage de canal. Elles augmentent la flexibilité des systèmes séparés et sont efficaces même si les propriétés du canal ne sont pas parfaitement connues de l'encodeur, ce qui est le cas si le canal n'est pas stationnaire.

Ces remarques permettent de formuler les domaines d'applications possibles des différents types de schémas de transmission. Le codage de source robuste n'impose pas de contrainte sur la structure du système de transmission, puisque celui-ci peut rester séparé. Les systèmes qui utilisent du codage conjoint requièrent des conditions d'utilisation strictes qui tendent à limiter leur utilisation à des schémas de transmission intégrés, tels les systèmes de communication. Le décodage conjoint source/canal se positionne comme une solution intermédiaire, puisqu'il peut être utilisé dans un certain nombre de solutions standardisées où un terminal doit effectuer le décodage de source et de canal, comme la télévision par satellite ou la téléphonie mobile.

2.3 Techniques de codage de source robuste et source/canal

Dans cette partie, nous nous proposons de présenter une liste non exhaustive de techniques de codage et décodage conjoint source/canal et soulignons le lien entre ces techniques et les contributions de cette thèse. Nous ne décrirons pas les techniques nécessitant un canal de retour, telles que les techniques très populaires de retransmission. Nous n'évoquerons pas non plus les techniques qui s'appliquent principalement aux canaux à effacements, même si certains types de description multiple, comme les codes BCH sur $\mathbb{R}$, permettent également de corriger ces erreurs [Bla79][WS95][Gab01]. Rappelons que l'objectif de cette thèse est de rendre robuste les trains compressés sur les canaux à erreurs. C'est pourquoi cette partie se concentre sur les techniques de décodage conjoint source/canal à base de codage entropique et évoque quelques techniques efficaces de codage de source robuste.

2.3.1 Impact des erreurs sur les codes de source

Les codes à longueur variable, largement utilisés dans les schémas et normes modernes de compression, sont extrêmement sensibles aux erreurs bits, même si leur nombre est très limité. Considérons par exemple le code de la figure 2.2 proposé dans [MR85] et l'automate de décodage associé. En étendant l'exemple proposé dans [MR85], nous pouvons observer sur les figures 2.3 et 2.4 qu'une simple erreur bit peut entraîner plusieurs erreurs de reconstruction des symboles. Remarquons que le décodeur finit par émettre les mêmes symboles que les symboles sources émis : le décodeur se resynchronise. Des méthodes caractérisant les codes qui contiennent une chaîne garantissant la resynchronisation du code ont été proposées pour une large classe de sources [FR84]. Les propriétés de resynchronisation des codes à longueur

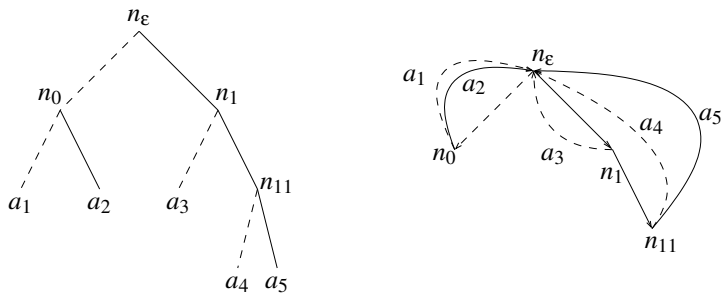


FIG. 2.2 – Exemple de code à longueur variable et automate de décodage correspondant (code proposé dans [MR85]).

variable en présence d'une erreur bit unique sont désormais bien connues. Elles ont été étudiées d'une manière générale dans [MR85] sous le nom de *error recovery (for variable length codes)*. Dans cet article[2], les auteurs étudient une mesure appelée *error span*, qui correspond à l'espérance du nombre de symboles sources émis avant que le décodeur resynchronise avec le codeur. Cette resynchronisation signifie que les états internes du codeur et du décodeur sont identiques. Ces états correspondent aux états de l'automate de la figure 2.2. La valeur de l'error span est généralement finie. Nous reviendrons sur la manière dont cette valeur est calculée dans le chapitre 6.

Bien que les deux exemples (Fig. 2.3 et 2.4) soient très similaires, ils se distinguent par le nombre de symboles décodés. En effet, sur la figure 2.3, le nombre de symboles décodés correspond au nombre de symboles émis. En revanche, sur la figure 2.4, le décodeur ne décode pas le même nombre de symboles. Dans les deux cas, le codeur a atteint l'état de resynchronisation. Cependant, cette notion de resynchronization occulte le fait que dans le second exemple, les symboles sont reconstruits de manière décalée [3]. Ainsi, nous distinguerons dans la suite les deux types de resynchronisation suivants :

1. *La resynchronisation forte*[4] [KN00] indique qu'il n'y a pas de décalage entre le codeur et le décodeur.
2. *La resynchronisation faible* (*weak synchronisation* dans [KN00]) est la resynchronisation prise au sens du noeud interne dans l'arbre de décodage, qui correspond souvent à l'état interne du décodage [MR85].

Lorsqu'aucune précision n'est donnée, la signification de resynchronisation correspond par défaut à la resynchronisation faible. La resynchronisation forte correspond à une resynchronisation des séquences au sens de la distance de Hamming, alors que

[2] Précisons que cet article contient quelques erreurs, corrigées dans [ML87].
[3] ce phénomène est souvent appelé *slippage*.
[4] *synchronisation with timing* ou *strong synchronization*

la resynchronisation faible correspond à une resynchronisation au sens de la distance de Levenshtein [Lev66]. Cette distance est souvent appelée aussi la distance d'édition-insertion. Il en existe des variantes mais la définition la plus largement répandue est la suivante : c'est le nombre minimum d'opérations de substitution, d'insertion et de suppression à effectuer pour transformer une séquence en une autre. Elle peut être calculée en appliquant l'algorithme de Wagner et Fischer [WF74], qui est un algorithme de programmation dynamique dont le coût de calcul est quadratique par rapport à la longueur de la séquence.

La plupart des travaux qui sont reliés à cette problématique consistent

- à analyser, avec différents formalismes, les mesures de résistance aux erreurs [MR85][ZZ02],
- à introduire des nouvelles mesures de résistance aux erreurs [SD95][Tit97][ZZ02],
- à trouver des algorithmes optimisant les valeurs de ces mesures [TWM94][ZZ02],
- à étendre l'analyse aux erreurs multiples et à des canaux particuliers, comme le canal binaire symétrique dans [TWM94] ,
- à utiliser et analyser l'impact de mots de code synchronisant [FR84][PSR04].

L'analyse des propriétés de resynchronisation des codes en fonction des mesures de performance des CLV a permis de mettre au point de "bons" codes en termes de resynchronisation, avec un intérêt plus particulier donné aux codes minimisant le délai de resynchronisation (faible), c'est-à-dire l'*error span*.

La mesure dite de *gain/loss* va également nous intéresser. Son calcul est proposé dans [SD95] et rappelé dans le chapitre 7. Elle mesure la différence, positive ou négative, entre le nombre de symboles émis et le nombre de symboles décodés. Ainsi, lorsque la valeur du *gain/loss* est égale à 0, cela signifie que la séquence se resynchronise au sens de Hamming, et réciproquement.

Le phénomène est considéré plus en détail sur les figures 2.5 et 2.6. Ces figures représentent les treillis qui correspondent au dépliage, par rapport au temps, des automates de décodage. Les transitions qui correspondent au bit 0 sont dessinées en pointillés et celles qui correspondent au bit 1 sont en traits plein.

Sur ces exemples, il apparaît que le décodeur se resynchronise au bout d'un certain temps avec l'encodeur, ou plus précisément il se retrouve dans le même état qu'un décodeur qui aurait reçu une version non bruitée du train binaire. Cependant, ces exemples diffèrent par le nombre de symboles décodés, qui sont indiqués en rouge sur la figure. Dans le cas de l'exemple d'erreur de la figure 2.5, le nombre de symboles décodés est identique au nombre de symbole effectivement encodés, c'est-à-dire que la resynchronisation est forte. *A contrario*, dans le cas de l'exemple de la figure 2.6, le décodeur a décodé un nombre inférieur de symbole. Le décodage résultant est très mauvais pour les mesures d'erreur objectives comme le taux d'erreur symbole ou l'erreur quadratique moyenne. En effet, dans le cas d'une source sans mémoire, la séquence

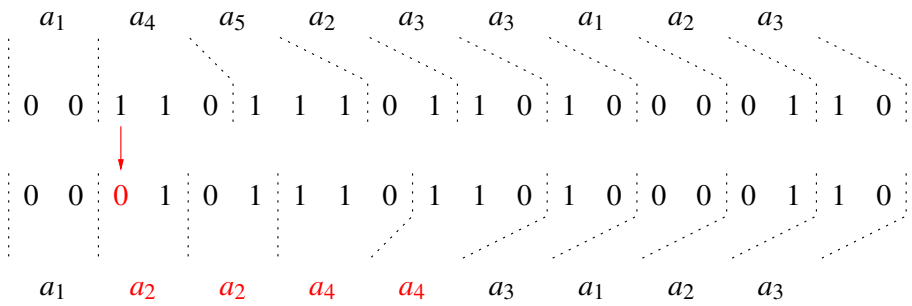


FIG. 2.3 – Impact d'une erreur sur le décodage : désynchronisation du décodeur avec resynchronisation au sens de la distance de Hamming (resynchronisation forte).

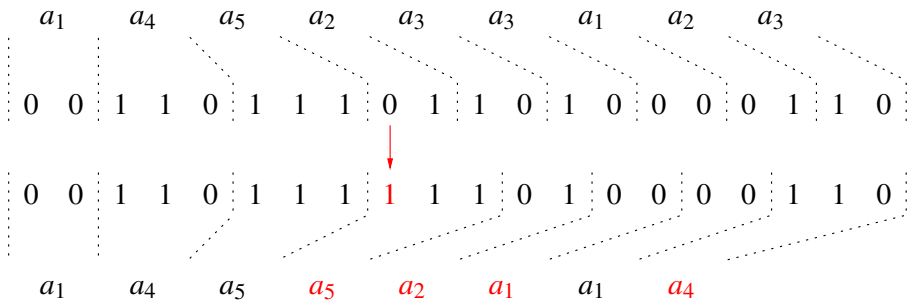


FIG. 2.4 – Impact d'une erreur sur le décodage : désynchronisation du décodeur sans resynchronisation au sens de la distance de Levenshtein (resynchronisation faible).

décodée a les propriétés d'une séquence aléatoire tirée selon la source, sans relation avec la séquence émise.

Le lien entre les propriétés de résistance aux erreurs des codes de source apparaît dans ce manuscrit de la manière suivante. Dans le chapitre 3 et 4, des nouveaux codes sources sont introduits. Ils *garantissent* une synchronisation stricte à un sous-ensemble des données considéré comme prioritaire, c'est-à-dire les données qui ont le plus fort impact sur la reconstruction. Dans le chapitre 5, l'analyse de ces propriétés est utilisée pour quantifier l'information qu'une contrainte de terminaison peut offrir à un estimateur. Elle motive ainsi l'approche de décodage souple proposée dans ce même chapitre en montrant que cette quantité d'information est peu affectée par la technique d'agregation d'états proposée.

2.3.2 Techniques de paquétisation

Étant donné que la propagation d'erreur est un phénomène catastrophique, des auteurs ont proposé de contrer ce problème en utilisant des marqueurs de resynchro-

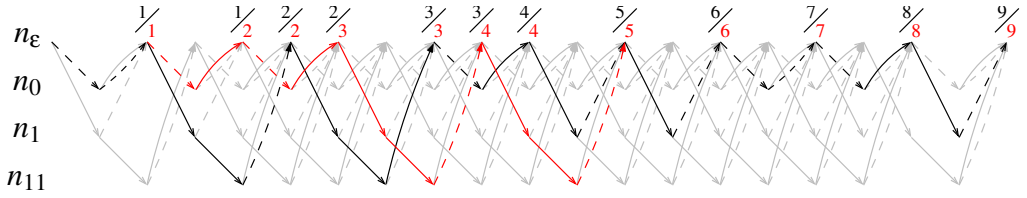


FIG. 2.5 – Impact d'une erreur sur le décodage en treillis : resynchronisation forte.

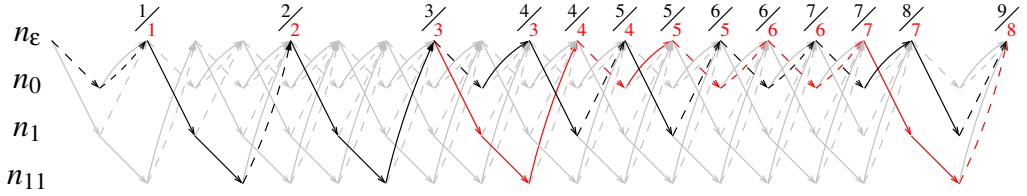


FIG. 2.6 – Impact d'une erreur sur le décodage en treillis : resynchronisation faible.

nisation (voir par exemple [KDKM00]) ou en intégrant des données d'aide à la resynchronisation dans les standards, comme le data-partitioning [LKH+00].

Le codeur entropique résistant aux erreurs (EREC : Error-Resilient Entropy Coding) proposé dans [RK96] et amélioré dans [WK99] peut également être considéré comme une technique qui effectue la paquétisation conjointement avec le codage entropique. La structure d'encodage résultante est une structure qui contient plusieurs blocs de petite taille, comme illustré sur la figure 2.7. Les données à transmettre sont groupées par blocs de longueur variable. La fin des blocs les plus longs vient compléter les blocs les plus courts en un nombre fini d'étapes. Ces étapes d'égalisation des longueurs de bloc peuvent être vues comme une paquétisation particulière du train binaire. Cette stratégie diminue l'incertitude du positionnement des bits dans le paquet et a pour effet de limiter considérablement l'impact d'une erreur sur la reconstruction. En effet, cette paquétisation offre des points de resynchronisation qui permettent d'avoir un décodage indépendant des codes préfixes sur le début des blocs. Ces points de resynchronisation sont définis de manière implicite et requièrent une quantité d'information supplémentaire qui peut être négligée. Dans le chapitre 6, nous proposons un ensemble de techniques dans la lignée du codeur entropique résistant aux erreurs. Le principal avantage des solutions proposées par rapport à la solution pré-existante est que l'encodage (et le décodage) peuvent être effectués en temps linéaire par rapport à la longueur de la séquence, avec un coût algorithmique très proche de l'encodage concaténé usuel.

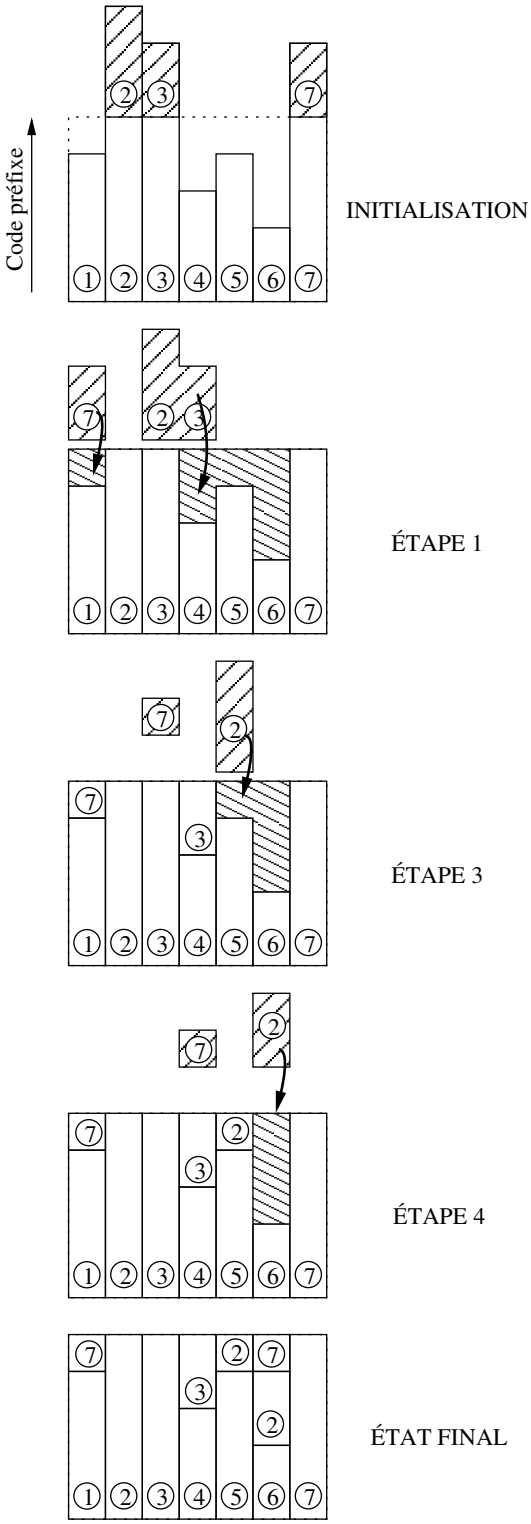


FIG. 2.7 – Codage entropique résistant aux erreurs : techniques de paquétisation avec points de resynchronisation implicites.

2.3.3 Codes réversibles

Les codes à longueur variable habituellement utilisés et en particulier les codes de Huffman sont des codes dits prefix-free, c'est-à-dire que la "lecture" à partir du début du mot permet de décoder de manière non ambiguë les symboles. Il n'est en revanche pas possible en pratique de décoder dans l'autre sens, car les codes de Huffman ne sont pas suffix-free. Si l'on considère par exemple le code $\{0, 10, 11\}$ et que la séquence à décoder se termine par les bits 1110, il n'est pas possible de décider si on a le mot de code 11 suivi du mot de code 0 ou si il s'agit au contraire du mot de code 11 suivi du mot du code 10.

Il est cependant possible de construire des codes qui admettent un décodage instantané dans les deux sens. Ces codes, introduits dans [TWM95], sont appelés des codes à longueur variable réversibles (CLVR), car ils admettent deux arbres de décodage, un pour un sens et un autre pour l'autre sens. Si ces deux arbres sont identiques, le code réversible est dit symétrique.

Le décodage bi-directionnel, en permettant de décoder le train binaire dans les deux sens, peut identifier un segment erroné. En effet, les codes CLVR ne sont pas complets : leur somme de Kraft est strictement inférieure à 1. En présence d'erreur, une détection d'un mot de code impossible permet de conclure qu'il y a un bit en erreur ou plus dans le train binaire. Pour certains de ces codes, une erreur bit simple introduit systématiquement un mot de code invalide au décodage et est toujours détectée.

L'inconvénient inhérent à ces codes est qu'ils ne compressent pas au mieux. Ils introduisent une redondance souvent supérieure à 10% par rapport à un code de Huffman, ce qui est loin d'être négligeable. Ils sont cependant largement considérés dans la communauté de décodage conjoint, car la propriété de détection des erreurs simples les rend très robustes au bruit lorsqu'ils sont décodés avec un décodage souple tel que Viterbi. C'est pourquoi des chercheurs continuent de proposer des constructions de codes réversibles menant à la plus petite espérance de longueur possible [TW01]. Nous les utiliserons également dans le chapitre 5.

2.3.4 Étiquetage binaire optimisé canal

Afin d'obtenir les meilleurs performances débit/distorsion, les mots binaires définissant l'index d'une cellule de quantification doivent être définis de manière à ce qu'une erreur sur un bit entraîne une distorsion qui est la plus faible possible, en espérance. Ce problème est celui de l'étiquetage binaire (*Index assignment*).

C'est un problème difficile pour lequel il est nécessaire d'avoir recours à des techniques d'optimisation. Un type d'optimisation efficace proposé dans [ZG90] pour ce problème est une variante de la descente de gradient et procède par permutation d'un bit à la fois. Cet algorithme est appelé *binary switching algorithm*. Il est aussi connu sous le nom de *pseudo-Gray coding*. Cependant, cette technique ne permet pas d'éviter les

extrema locaux. Pour éviter ce type d'extremum, les techniques de recuits simulés ont été montrées efficaces pour optimiser les codes [GHSW87]. Elles ont été appliquées à la quantification optimisée canal dans [FV91] où il est montré que, non seulement les quantificateurs obtenus sont meilleurs, mais la convergence est rapide. Ce type de technique a été appliqué au codage d'image dans [TF92] et dans [PSA98], où les auteurs proposent l'optimisation de la quantification vectorielle conjointement avec la modulation canal.

Bien que le problème de l'étiquetage binaire ait été principalement considéré dans un contexte de quantification vectorielle, les techniques d'optimisation qui s'y rapportent peuvent s'appliquer à d'autres problèmes d'étiquetage binaire. C'est pourquoi nous les avons considérées pour l'optimisation des partitions des dictionnaires dans le chapitre 4.

2.3.5 Protection inégale aux erreurs

Les signaux naturels contiennent des informations dont l'importance pour la reconstruction est très inégale. Ainsi, la plus grande partie de l'énergie utile d'un signal est contenue dans très peu de bits, comme en témoignent les excellentes performances en compression atteintes par les codeurs en sous-bandes. En revanche, pour raffiner l'information, il faut transmettre un nombre de bits important. Lorsqu'un tel signal doit être transmis sur un canal bruité, il paraît naturel d'offrir une meilleure protection aux bits qui contiennent la majorité de l'énergie de reconstruction. Les techniques qui exploitent la non uniformité de l'importance des coefficients de source sont dites *de protection inégale* (UEP : *Unequal Error Protection*). Beaucoup des techniques proposées reposent sur l'utilisation de codes convolutifs décimés [Hag88] (RCPC : *Rate Compatible Convolutional Codes*). Dans [WK99] les auteurs suggèrent d'utiliser le codeur entropique résistant aux erreurs de [RK96] pour offrir un niveau de résistance inégale aux erreurs selon la position dans le train binaire. Dans les chapitres 3 et 4, nous proposons un système dont le but est d'offrir une résistance inégale au niveau du codeur de source. À notre connaissance, c'est la première fois qu'un système de protection inégale agissant au niveau du codage entropique est proposé.

2.3.6 Décodage souple de CLV

Sauf pour des cas singuliers tels que les sources diadiques, le codage entropique à base de codes de Huffman est sous-optimal (les performances en compression sont inférieures à l'entropie du signal). C'est encore plus vrai pour les CLVR ou les codes à longueur variable correcteurs d'erreur [BF93][But95]. La redondance résiduelle peut donc *a priori* être exploitée pour améliorer la reconstruction en présence de bruit sur le canal [SB91]. Ceci est également vrai pour d'autres types de code de sources. Ainsi, l'exploitation de la redondance résiduelle a également été considérée pour l'algorithme LZ77, en particulier dans [LS03], et pour les codes arithmétiques [Say99][GG03]

et quasi-arithmétiques [GG04].

Le premier à avoir proposé un modèle d'état pour le décodage souple des CLV est Balakirsky [Bal97]. Il propose de voir le décodage comme un problème d'estimation d'une chaîne de Markov cachée où la variable à estimer est l'état interne d'un décodeur qui aurait reçu correctement les bits. Ce problème a ensuite été décliné sous plusieurs variantes. Les hypothèses majoritairement retenues sont les suivantes :

- La source est considérée invariante, discrète, sans mémoire ou markovienne (d'ordre 1) [MF99][MF98a][SV99][Wei03][TK03].
- Le nombre de bits émis est le plus souvent connu. Il est possible de considérer que le nombre de symboles uniquement est connu [GFGR01]. Si les deux informations sont connues comme dans [SV98][MF99][BH00a][GFGR01], le décodage sera dit *avec contrainte de longueur*.

Différents types de décodage s'appliquent alors (BCJR, Viterbi ou List-Viterbi). Dans le cas où le nombre de symboles est connu au décodage et dans le cas où le système d'encodage est composé d'un CLV et d'un code convolutif, Murad et Fuja proposent et comparent dans [MF00] plusieurs méthodes de décodage sous-optimales : décodage par liste, élagage du treillis. L'approche qui donne les meilleurs résultats dans leur simulation est le décodage par liste modifié. Ainsi, le décodage de canal est effectué de manière à ce que la longueur des séquences soit prise en compte au moment du choix des L meilleures séquences : en effet, pour chaque état, le décodeur conserve les L meilleurs chemins de longueurs distinctes au lieu des L meilleurs chemins dans une des autres approches à base de décodage par liste. Le gain sur le canal par rapport à l'approche séparée est d'environ 1 dB.

Alors que la désynchronisation des CLV est considérée comme un désavantage de ces codes, les auteurs montrent dans [SOD00] que les performances obtenues en décodage souple sont meilleures avec un CLV qu'avec un CLF lorsque le taux d'erreur est petit. Ainsi, le code CLV possède une capacité intrinsèque de correction d'erreur.

Dans le chapitre 5, nous considérons le problème du décodage souple lorsque qu'une contrainte de longueur est disponible et proposons une solution d'agrégation d'état qui réduit très significativement la complexité de l'estimation. Contrairement aux techniques dites de *pruning* [MF00][KT05] qui introduisent une sous-optimalité par rapport au décodage optimum, notre solution n'effectue pas d'élagage et atteint les mêmes performances que le décodage optimum pour une complexité linéaire par rapport à la taille de la séquence.

2.3.7 Décodage conjoint source-canal de CLV

L'engouement provoqué par les turbo-codes [BGP93] a amené des auteurs à considérer l'application du principe turbo au décodage conjoint source/canal à base de CLV. Ainsi, les auteurs dans [BH00d] proposent d'exploiter la redondance résiduelle

des CLV en utilisant en série un code CLV, d'un entrelaceur et d'un code correcteur d'erreur convolutif. Le schéma résultant correspond à un turbo-code série où le premier code convolutif est remplacé par un CLV.

De nombreuses approches similaires ont été proposées. Parmi celles-ci, on peut citer un schéma ayant des similitudes avec un turbo-code parallèle [KT03] ou bien l'utilisation de codes turbo en série après le CLV [GCS00][GFGR01][JV04][JCS05].

Les techniques que nous proposons dans cette thèse peuvent s'appliquent encore dans un cadre turbo. En effet, l'adjonction d'un turbo-code en série avec un estimateur souple de source pour former un super code source/canal est un procédé bien connu qui a montré son efficacité. Dans cette thèse, ce type de système a été considéré uniquement à des fins de comparaison, en particulier dans le chapitre 6.

2.4 Résumé des contributions

2.4.1 Codes multiplexés

Le chapitre 3 présente les codes multiplexés et correspond à une version longue de l'article de journal [JG05d]. Il a été présenté en partie en conférence [JG03a][JG03d] et dans un rapport de recherche [JG03c]. Selon notre classification introduite en 2.2, les codes multiplexés sont une technique de codage de source robuste, ce qui leur offre de ce point du vue une grande souplesse car ils peuvent être utilisés même lorsque le système de transmission est séparé.

Les codes multiplexés sont des codes entropiques qui se proposent de supprimer la désynchronisation des codes à longueur variable de manière structurelle. Ils utilisent le fait que les systèmes de compression réels génèrent des sources d'information avec des niveaux de priorité différents. Par exemple, l'information de mouvement est plus importante que celle de texture en vidéo. Par *plus importante*, nous signifions du point de vue de l'erreur de reconstruction, généralement mesurée par l'erreur quadratique moyenne. La motivation sous-jacente est donc la même que celle des techniques de protection inégale aux erreurs. La différence principale avec les techniques de protection inégale classiques réside dans le fait qu'aucune redondance n'est introduite. Il est donc plus précis dans ce contexte de qualifier la technique de *résistance inégale aux erreurs*.

La construction de ce type de code se base sur un code à longueur fixe redondant. L'ensemble des mots de code est ensuite partitionné en autant de *classes d'équivalence* qu'il y a de symboles dans l'alphabet de la source discrète. Ainsi, à chaque mot de code de ce dictionnaire est associé un symbole de la source prioritaire et une valeur d'index qui permet de stocker des informations de la source moins prioritaire. Cette dernière valeur d'index peut être vue comme la réalisation d'une variable aléatoire dont la valuation est conditionnée par la réalisation de la source haute priorité. La technique est schématisée sur la figure 2.8.

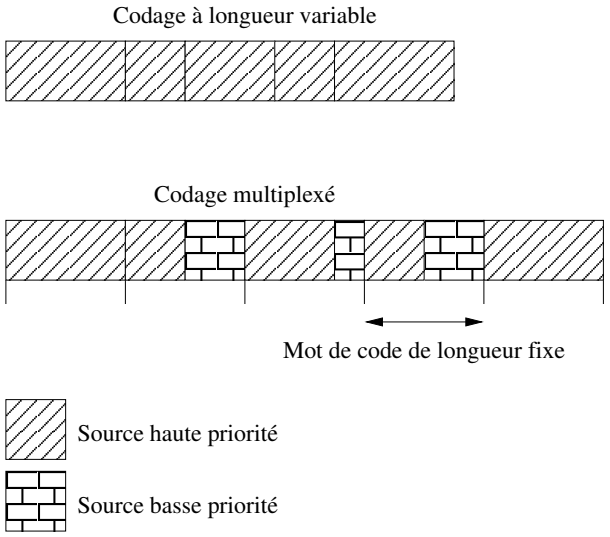


FIG. 2.8 – Représentation schématique du principe des codes multiplexés. En haut, les mots de code à longueur variable sont sujet à désynchronisation. En bas, les codes multiplexés garantissent la synchronisation de la source haute priorité.

La source de basse priorité est supposée encodée avec un algorithme de compression optimal. Dans ce cas, le train binaire généré est une suite de réalisations binaires (0 et 1) aléatoires et uniformes. La performance en compression de ces codes est analysée en deux parties. Tout d'abord, pour la source haute priorité, il est démontré qu'en augmentant la longueur des mots de code, il est possible d'être arbitrairement proche de l'entropie d'une source sans mémoire. Cette augmentation se fait au détriment du taux de source prioritaire. La question de ce compromis est abordée et il est montré qu'une vectorisation de la source permet d'améliorer ce compromis au coût d'une augmentation de la taille du code.

La question de la conversion de la source (binaire) de moindre importance en une suite de variables dont la valuation est conditionnée par la source haute priorité est alors traitée. Un algorithme presque linéaire, la décomposition euclidienne hiérarchique, est proposé à cet effet. La sous-optimalité d'une telle conversion est majorée par 1 bit pour l'ensemble de la séquence. Bien que cette décomposition repose sur de l'arithmétique de nombre entiers en précision arbitraire, nous démontrons que la complexité algorithmique reste -relativement- faible. Cette analyse est confirmée par des mesures en simulation du coût de calcul. Pour garantir la linéarité stricte des processus d'encodage et de décodage, une autre technique de conversion de train binaire est proposée. Cette technique se base sur la décomposition d'une variable aléatoire en plusieurs variables aléatoires de plus faible cardinal. Elle introduit la contrainte suivante sur la partition : la décomposition en facteurs premiers des cardinaux des

classes d'équivalence ne doit contenir que des entiers qui sont inférieurs ou égaux à un petit nombre premier fixé, par exemple 5 ou 7. Par comparaison à une partition non contrainte, cette contrainte introduit donc une sous-optimalité dans l'approximation de la loi de la source. Par ailleurs, la conversion proposée de ces variables élémentaires de faible cardinal en bits introduit également une sous-optimalité. Celle-ci est, en pratique, bornée et inférieure à 1% de débit supplémentaire (*overhead*).

La résistance aux erreurs est alors analysée à la fois sur une source théorique et sur une source réelle représentée ici par une image. L'analyse de la source théorique montre que la source de haute priorité est plus résistante lorsqu'elle est encodée avec des codes multiplexés plutôt qu'avec des codes à longueur fixe.

Ces codes ont été étendus de manière à exploiter les statistiques d'ordre supérieur de la source en considérant que celle-ci est une chaîne de Markov. Cette extension est décrite dans le chapitre 4. Ce chapitre correspond à un article de journal [JG03b] et a été d'abord été présenté dans [JG04c] . Dans cet article, la construction des codes est tout d'abord abordée pour atteindre l'entropie d'ordre 1 de la source. L'analyse est relativement classique et consiste à créer, pour chacune des valeurs à l'instant $t-1$ conditionnant la loi à l'instant t, un code multiplexé avec les algorithmes décrits dans le chapitre 3.

Le problème de la résistance aux erreurs est en revanche plus ardu. Tout d'abord, nous proposons l'analyse de la résistance aux erreurs pour un canal discret sans mémoire. L'analyse repose sur l'analyse de l'état stationnaire de la chaîne de Markov correspondant au couple (mot de code émis, mot de code reçu). Nous proposons donc le calcul de la matrice de transition équivalente, ce qui nous permet d'obtenir la probabilité marginale comme un vecteur propre particulier de cette matrice : celui qui correspond au seul vecteur propre positif associé à la valeur propre de module 1 (qui vaut exactement 1 dans notre cas car la chaîne est supposée ergodique). La probabilité stationnaire du couple émis/reçu permet alors de déduire l'espérance du nombre de symboles de la source prioritaire en erreur, menant au taux d'erreur symbole. Nous l'utilisons également pour en déduire l'erreur quadratique moyenne.

Il est intéressant d'obtenir ces mesures de cette manière (analytique) pour deux raisons. Premièrement, ils sont plus précis que les résultats de simulation, avec lesquels ils s'accordent bien. Mais surtout, cela permet l'optimisation de ces codes par des techniques classiques. Nous avons en particulier considéré l'optimisation basée sur le recuit simulé (*simulated annealing* en anglais). Cependant, nous avons constaté que la convergence vers une bonne solution ne permettait pas toujours d'éviter les minima locaux, dépendait beaucoup de l'état initial, et surtout pouvait prendre beaucoup de temps, car la taille de l'espace des codes considérés est très grande.

C'est pourquoi nous avons proposé une optimisation en deux étapes. La première étape est une optimisation qui ne dépend pas du canal et qui consiste à minimiser la probabilité qu'un contexte soit en erreur. En effet, le caractère adaptatif de la partition

rend plus vulnérable la reconstruction, car même si le mot de code reçu est correct, son interprétation peut être erronée, ce qui peut *a priori* conduire à des effets de propagation catastrophique. L'optimisation dans cette première étape est effectuée avec un algorithme d'optimisation non stochastique, car nous avons observé qu'un recuit simulé n'apportait rien dans ce contexte. La seconde étape consiste alors à optimiser l'étiquetage binaire par permutation des mots de code deux à deux. Ce procédé d'optimisation en deux étapes a une complexité raisonnable et mène à des bonnes performances globales. Selon la mesure de performance considérée, il peut cependant s'avérer intéressant de terminer l'optimisation par un algorithme de recuit simulé non contraint par le type de transformation valide et qui cherche une solution dans la totalité de l'espace des codes.

Les performances en résistance aux erreurs sont positives. En effet, en dépit du caractère adaptatif des partitions, les codes d'ordre 1 obtenus pour des sources quelconques sont très résistants aux erreurs et sont proches des performances d'erreur des codes à longueur fixe. Cela s'explique par la présence de mots de code qui sont décodés avec le même symbole de la source prioritaire quel que soit le contexte. L'ensemble de ces mots de code est appelé le noyau du code multiplexé et il apparaît que son cardinal est maximisé par l'algorithme d'optimisation du taux d'erreur séquence. En effet, la présence de nombreux éléments dans le noyau augmente la probabilité que l'état du décodeur se resynchronise quel que soit le passé. Il permet également un accès aléatoire sur lequel nous n'avons pas insisté.

Les codes multiplexés, contextuels ou non, ont alors été considérés dans un contexte de décodage souple utilisant une estimation bayésienne. La complexité de celui-ci est analysée et est relativement faible, en particulier par rapport aux algorithmes de décodage souple de CLV qui intègrent une contrainte de terminaison. Afin d'améliorer le décodage souple, l'insertion de mots de code sans mémoire (non conditionnés) à intervalle régulier a été proposée pour supprimer la dépendance avec le passé. Le compromis entre la sous-optimalité introduite en compression et le gain en résistance aux erreurs est alors étudié et peut être dosé par la fréquence d'insertion des mots de code sans mémoire.

2.4.2 Décodage conjoint source/canal

Le chapitre 5 présente les travaux effectués sur le décodage souple de codes à longueur variable. Ce travail a été effectué conjointement avec Simon Malinowski, stagiaire de Master recherche dont j'ai assuré l'encadrement. Il a donné lieu à deux publications de conférence [JMG05b][JMG05a] et fait actuellement l'objet d'une soumission en journal [MJG05].

Ce travail fait un lien entre les propriétés de resynchronisation des CLV et leurs performances en décodage souple lorsqu'une contrainte de terminaison est disponible au décodage. Contrairement à [JV05] qui utilise les outils d'analyse des codes correc-

teurs pour quantifier les performances des codes à longueur variable utilisés dans un contexte de décodage itératif, notre analyse ne mesure que l'effet de la contrainte de terminaison. En particulier, elle nous permet de mesurer

- la quantité d'information que cette contrainte fournit au décodeur ;
- la probabilité que la contrainte permette d'écarter une séquence qui n'aurait pas le bon nombre de symboles.

Une nouvelle technique de décodage souple de codes à longueur variable est alors proposée lorsque la contrainte de longueur est connue au décodage. Cette technique de décodage exploite le fait que les CLV resynchronisent rapidement. Elle repose sur un principe d'agrégation d'états du treillis de décodage qui permet de régler le compromis entre la précision de l'estimation et la complexité. Ainsi, des états éloignés peuvent être agrégés en un seul super-état. L'éloignement des états composant un super-état garantit que l'association super-état $\rightarrow$ état n'est pas ambiguë si un décodage au sens du MAP est effectué sur le treillis des super-états.

Ces travaux peuvent être rapprochés de ceux de [MKLKD05] et [DW05]. En effet, dans ces articles les auteurs proposent également des techniques de diminution du nombre d'états. Ainsi, les auteurs proposent dans [MKLKD05] de réduire le nombre d'états pour les arbres CLV qui ont une taille importante. La diminution de complexité associée est considérée dans le contexte du décodage souple des données de texture d'un codeur vidéo. Dans [DW05], les auteurs montrent qu'il est possible de réduire significativement le nombre d'états du super-treillis associé au couple décodeur de source/décodeur de canal.

Dans [ND03], les auteurs proposent de quantifier la redondance existante au sein d'un code à longueur variable. Cette approche est poussée plus loin dans cette thèse. En effet, l'approche de Hang ne prend pas complétement en compte les propriétés du code source et n'inclut pas le canal. De ce fait, si la redondance présente est mesurée, elle ne correspond pas exactement à la quantité d'information *exploitable* par le décodeur à longueur variable. C'est cette quantité que notre analyse vise à mesurer, et en particulier pour la contrainte de terminaison du treillis agrégé. La quantification de la quantité d'information présente montre que les performances du modèle d'état optimum peuvent être obtenues avec un treillis de faible complexité. Cette analyse est confirmée par les résultats de simulations, qui montrent que pour un paramètre d'agrégation bien choisi, les performances du treillis optimal (le treillis bit/symbole) sont atteintes, et ce, à un coût de calcul bien inférieur. Ainsi et contrairement aux techniques d'élagage (*pruning*) qui introduisent une sous-optimalité, les performances optimales du treillis bit/symbole sont obtenues avec une complexité raisonnable. Une technique de "décodage combiné multi-treillis" est alors proposée pour réduire davantage l'espérance de la complexité de décodage. L'approche est particulièrement efficace lorsque le rapport signal à bruit est important. Elle permet d'adapter la complexité du décodage au bruit effectif sur le canal.

2.4.3 Construction de train binaire

La troisième contribution de cette thèse repose sur le constat que la concaténation des mots de code générés par un CLV n'est pas requise et peut être avantageusement remplacée lorsque la contrainte de délai est un peu relâchée (ce qui est d'ailleurs également le cas si un code correcteur de type turbo est utilisé). Les étapes de codage et de paquétisation peuvent ainsi être considérées séparément. C'est pourquoi nous proposons de remplacer la concaténation des mots de code issus des CLV par des algorithmes de construction du train binaire qui permettent d'augmenter la résistance aux erreurs bits. Ces techniques se rapprochent d'une technique de paquétisation connue sous le nom de codage entropique résistant aux erreurs [RK96]. Notre approche propose un cadre pour ce type de technique qui permet de considérer de nombreuses constructions de train binaire différentes. Les schémas obtenus offrent des performances similaires au codeur entropique robuste mais avec une complexité algorithmique plus faible. Par ailleurs, les schémas considérés peuvent être avantageusement couplés avec une construction appropriée du train binaire qui permet de concentrer l'énergie du signal sur les transitions bits qui sont moins affectées par l'effet de propagation des erreurs. Nous montrons aussi que de telles techniques de construction de train binaire permettent parfois d'améliorer la progressivité de reconstruction.

2.4.4 Nouveaux codes de source

La quatrième et dernière contribution de cette thèse consiste à utiliser des règles de ré-écriture d'une forme simple pour étendre les codes entropiques à longueur variable à base d'arbres binaires. La forme de ces codes est la suivante

$$a\bar{l} \rightarrow \bar{b},$$

où a est un symbole de l'alphabet à encoder et où $\bar{l}$ et $\bar{b}$ sont de petites séquences de bits. Cette définition permet d'obtenir des codes offrant des propriétés intéressantes. Tout d'abord, il est possible de définir des codes qui permettent de coder les symboles avec moins d'un bit, et ceci sans qu'il soit nécessaire de considérer l'alphabet produit. Ceci est en particulier montré par l'analyse des propriétés de compression du code.

Nous proposons ensuite deux techniques de construction à base de codes de Huffman. La première permet la construction d'un code lexicographique avec une faible espérance de longueur. Les codes obtenus sont comparés à un code de Hu-Tucker sur un exemple, qui sont les codes optimaux. Cependant, lorsqu'on veut encoder des séquences de symboles et non un symbole unique, cette optimalité n'est obtenue que par une vectorisation appropriée de la source, ce qui a pour effet de créer des codes ayant un nombre d'états qui croît comme une série géométrique avec la longueur de la séquence. C'est pourquoi dans un système pratique, la comparaison doit être effectuée sous la contrainte d'un nombre d'états maximum. C'est sous cette contrainte que

nous montrons que la construction proposée peut donner de meilleurs résultats que les codes de Hu-Tucker.

La deuxième méthode proposée construit un code tel que la probabilité marginale des bits tende vers l'uniformité, ce qui permet théoriquement d'obtenir de meilleures performances en décodage souple. Sa construction consiste à considérer le code dual du code originel, c'est-à-dire le code dont les mots de code sont les opposés au sens de Hamming. Le code originel et le code dual sont joints pour créer un code *miroir* ou un symbole est encodé soit avec un mot de code, soit avec le mot de code dual. L'intérêt de tels codes est vérifié en simulation lorsque le code est construit à partir d'un code réversible. Ce gain est obtenu gratuitement, c'est-à-dire que son espérance de longueur est identique à celle du code originel.

2.5 Conclusion

Cette partie a dressé un état de l'art des techniques de codage de source robuste et de codage/décodage conjoint source/canal. Celles-ci ont été classifiées en fonction de la possibilité de les utiliser pour un niveau d'intégration donné des codages de source et de canal. Les travaux présentés dans la seconde partie ont ensuite été décrits succinctement.

Deuxième partie

Contributions

Chapitre 3

Multiplexed codes

This chapter have been partially published in [JG05d] and [JG03c]. It has been presented in [JG03d].

Abstract : Compression systems of real signals (images, video, audio) generate sources of information with different levels of priority which are then encoded with variable length codes (VLC). This chapter addresses the issue of robust transmission of such VLC encoded heterogeneous sources over error-prone channels. VLCs are very sensitive to channel noise : when some bits are altered, synchronization losses can occur at the receiver. This chapter describes a new family of codes, called multiplexed codes, that allow the designer to confine the desynchronization phenomenon to low priority data while asymptotically attaining the entropy bound for both (low and high priority) sources. The idea consists in creating fixed length codes for high priority information and in using the inherent redundancy to describe low priority data, hence the name multiplexed codes. Theoretical and simulation results reveal a very high error resilience at almost no cost in compression efficiency.

3.1 Introduction

Entropy coding, producing variable length codewords, is a core component of any data compression scheme. Unlike fixed length codes (FLCs), variable length codes are designed to exploit the inherent redundancy of a symbol distribution. The main drawback of VLCs is their high sensitivity to channel noise : when some bits are altered by the channel, synchronization losses can occur at the receiver, the position of symbol boundaries are not properly estimated, leading to dramatic increases in symbol error rates. Soft VLC decoding ideas, exploiting residual source redundancy (the so-called "excess-rate"), have also been shown to reduce the "de-synchronization" effect as well as the residual symbol error rates. These ideas rely on capitalizing on source coder suboptimality, by exploiting inner codeword redundancy and exploiting correlation within the sequence of symbols (inter symbol dependency). Models incor-

porating both VLC-encoded sources and channel codes (CC) have also been considered [MF98b, DS98, BH00d]. The authors in [MF98b] derive a global stochastic automaton model of the transmitted bitstream by computing the product of the separate models (Markov source (MS), source coder (SC) and channel coder (CC)). The authors in [GFGR01] push further the above idea by designing an iterative estimation technique alternating the use of the three models (MS, SC, and CC). The above methods mostly consist in re-augmenting the redundancy of the bitstream, by introducing an error correcting code or dedicated patterns in the chain. Also, the decoding algorithms, often relying on MAP estimators, may have a rather high complexity if the termination constraint is to be exploited at the decoder (see Chapter 5).

The synchronization recovery capability (or self-synchronization property) of a VLC, which represents the error-resiliency of the code, has also been considered as a performance and design criterion of the code in addition to its compression performance. The characterization of the synchronization capability of VLC has thus been extensively studied in the research community [WS80], [CGV88]. This property is often characterized by an error span [MR85], also called the mean error propagation length (MEPL) [ZZ02]. It will be recalled with more details in Chapter 5. The authors in [MR85] have developed a state model for synchronization recovery which is suitable for analyzing the performance of various codes with respect to error recovery. Effort has then also naturally been dedicated to the design of self-synchronizing codes. In [FR84], the authors show that some Huffman codes, tailored to a given source, may contain a codeword that can serve as a "re-synchronization" point. However, the existence of this codeword depends on the source statistics. Statistically synchronizable codes containing a synchronization sequence are also described in [WS80] and [CSGV92]. Self-synchronizing Huffman codes are also proposed in [LR92]. The design is governed by a trade-off between redundancy and synchronization recovery capability. E.g., the authors in [TWM94] and [ZZ02] search to construct codes with minimum redundancy which have short mean error propagation length (MEPL). In the same spirit, reversible VLCs (RVLCs) [TWM95, WV98, BH00d] have been designed specifically to fight against desynchronizations. Variable-to-fixed length codes [Tun67] [SG97] mainly designed for compression purposes are also well-suited for robust transmission. Both self-synchronizing codes and variable-to-fixed codes only allow for synchronization in a Levenshtein distance sense [Lev66]. However, strict-sense synchronization [ZZ02], i.e. synchronization in the Hamming distance sense, is required by applications, such as image and video compression.

Here, we consider the design of a new family of codes, called "multiplexed codes", that allow one to control (even avoid) the "de-synchronization" phenomenon of VLC encoded sources, while still allowing one to reach asymptotically the entropy bound and to rely on simple decoding techniques. The principle underlying these codes builds upon the idea that compression systems of real signals generate sources of information with different levels of priority (e.g. texture and motion information

for a video signal). This idea underlies UEP techniques, which allocate different levels of protection to the different types of information, either to signal frequency bands [TF92], to bit planes [RM95], or to quantization layers [CZ96]. In the wake of this idea, we consider two sources, a high priority source and a low priority source referred to as $\mathbf{S}_H$ and $\mathbf{S}_L$ in the sequel. The codes designed are such that the risk of "de-synchronization" is confined to the low priority information. The idea consists of creating a FLC for the $\mathbf{S}_H$ source, and in exploiting the inherent redundancy to represent or store information of the low priority source $\mathbf{S}_L$. Hence, the source $\mathbf{S}_H$ inherits some of the properties of FLCs such as random access in the data stream and strict-sense synchronization. It is shown that these codes nearly attain the entropy bound.

The rest of the chapter is organized as follows. Section 3.2 introduces the notation we use in the sequel. Section 3.3 outlines the principle of multiplexed codes, discusses their compression properties and describes the first coding algorithm in the case of two sources. The approach relies on a mapping of the low priority flow of data into a sequence of k-valued variables, where k depends on the realization of the high priority sequence of symbols. A first mapping, optimal in the sense that it can attain the entropy bound is presented. This mapping is based on an Euclidean decomposition of a long integer, which can be computed with a hierarchical (dyadic) approach with a close to linear computational complexity. The computational cost can be further reduced (but this time at the expense of a small excess-rate), by performing the decomposition on a constrained set of k-valued variables. This amounts to consider a constrained partition of the set of fixed length codewords into *equivalence classes* expressed in terms of prime-valued variables. The resulting loss in compression depends on the distance between the respective probability mass function (PMF) of the constrained set of k-valued variable and of the source $\mathbf{S}_H$. A heuristic method for selecting the set of k-valued variables (i.e., the partition) that would best approximate the PMF of the source $\mathbf{S}_H$ is described in section 3.8. A second class of multiplexed codes for which the cardinalities of the different equivalence classes are constrained to be integer powers of two is also presented. Such codes can be built in a very straightforward manner from any VLC code. The compression performance is then the same as the one achieved by the VLC considered. The impact of channel noise, considering a binary symmetric channel, on the overall source distortion is analyzed in section 3.9. The choice of the parameters of the algorithm are discussed in section 3.10 and simulation results are provided in section 3.11.

3.2 Problem statement and notation

Let $\mathbf{S}_H = (S_1, \ldots, S_t, \ldots, S_{\mathbf{K}_H})$ be a sequence of source symbols of high priority taking their values in a finite alphabet $\mathcal{A}$ composed of $|\mathcal{A}|$ symbols, $\mathcal{A} = \{a_1, \ldots, a_i, \ldots, \}$. Note that, in the following, we reserve capital letters for random variables, and small

class $\mathcal{C}_i$	codeword x	symbol a_i	$\lvert\mathcal{C}_i\rvert$	probability μ_i	index q_i
$\mathcal{C}_1$	000	a_1	3	0.40	0
	001				1
	010				2
$\mathcal{C}_2$	011	a_2	2	0.20	0
	100				1
$\mathcal{C}_3$	101	a_3	1	0.20	0
$\mathcal{C}_4$	110	a_4	1	0.10	0
$\mathcal{C}_5$	111	a_5	1	0.10	0

TAB. 3.1 – An example of multiplexed codes ($c = 3$).

letters for values of these variables. Bold face characters will be used to denote vectors or sequences. The stationary PMF of the source $\mathbf{S}_H$ is denoted $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_i, \ldots, \mu_{|\mathcal{A}|})$, where μ_i stands for the marginal probability that a symbol of $\mathbf{S}_H$ equals a_i. Let h be the stationary entropy of the source per symbol, given by $h = -\sum_{i=1}^{|\mathcal{A}|} \mu_i \log_2(\mu_i)$. Let $\mathbf{S}_L$ be a sequence of source symbols of lower priority taking their values in a finite alphabet. We assume that the realization $\mathbf{s}_L$ of this source has been pre-encoded into a bitstream $\mathbf{b} = (b_1, \ldots, b_r, \ldots, b_{K_B})$ with a VLC coder (e.g. Huffman or arithmetic coder). The problem addressed here is the design of a family of joint codes for the two sources $\mathbf{S}_H$ and $\mathbf{S}_L$ that would guarantee the strict-sense synchronization of the high priority source $\mathbf{S}_H$ at almost no cost in terms of compression efficiency.

3.3 Multiplexed codes

3.3.1 Principle and definitions

Let us denote by $\mathcal{X}$ the set of binary codewords of length c. This set of $|\mathcal{X}| = 2^c$ codewords is partitioned into $|\mathcal{A}|$ subsets, denoted $\mathcal{C}_i$ for $i = 1 \ldots |\mathcal{A}|$, called *equivalence classes* associated with symbols a_i of the alphabet $\mathcal{A}$, as shown in Table 3.1. Note that codebook partitioning is sometimes referred to as binning. The condition $c \geq \log_2(|\mathcal{A}|)$ is required to have at least 1 codeword per subset. Each *equivalence class* $\mathcal{C}_i$ is a set of $|\mathcal{C}_i|$ codewords. In the following, the integer $|\mathcal{C}_i|$ denotes the cardinality of the *equivalence class* and is such that $\sum_{i=1}^{|\mathcal{A}|} |\mathcal{C}_i| = |\mathcal{X}|$.

A symbol $S_t = a_i$ of the flow $\mathbf{S}_H$ can be encoded with any c-bit codeword x belonging to the *equivalence class* $\mathcal{C}_i$ (see example of Table 3.1). Hence, each codeword S_t can be mapped into a pair $(\mathcal{C}_i, Q_i)$ of two variables denoting respectively the *equivalence class* and the index of the codeword in the *equivalence class* $\mathcal{C}_i$. The variable Q_i is an $|\mathcal{C}_i|$-valued variable, taking its value between 0 and $|\mathcal{C}_i| - 1$ (see Table 3.1) and representing the inherent redundancy of the c-bits FLC. This redundancy will be exploited

to describe the lower priority flow of data. Let us denote by n_t the cardinality $|\mathcal{C}_i|$ of the equivalence class associated with the realization $S_t = a_i$. Then, to the realization $\mathbf{s}_H = s_1 \dots s_t \dots s_{K_H}$ of the sequence of high priority symbols one can associate a sequence $q_1 \dots q_t \dots q_{K_H}$ of n_t-valued variables that will be used to describe jointly the high and low priority data flows (i.e., $\mathbf{s}_H$ and $\mathbf{s}_L$).

Définition 3.1 *A multiplexed code is defined as a set of pairs (a_i, q_i), where a_i is a symbol value belonging to the alphabet $\mathcal{A}$ (on which the high priority source is quantized) and where q_i is the value of a variable Q_i denoting the index of the codeword in the* equivalence class.

The corresponding encoder is the function which maps every couple (a_i, q_i) into an c-bits fixed length codeword x as

$$(a_i, q_i) \leftrightarrows x. \tag{3.1}$$

Example 3.1: Let $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$ be the alphabet of the source $\mathbf{S}_H$ with the stationary probabilities given by $\mu_1 = 0.4$, $\mu_2 = 0.2$, $\mu_3 = 0.2$, $\mu_4 = 0.1$ and $\mu_5 = 0.1$. Note that this source has been considered in [MR85] and [ZZ02]. Table 3.1 gives an example of partitioning of the set $\mathcal{X}$ into the 5 equivalence classes *associated with the alphabet symbols. This partitioning reveals* 5 *$|\mathcal{C}_i|$-valued variables.*

Note that, in Table 3.1, the codes are ranked in the lexicographic order. Using this order, each equivalence class can be fully defined by, 1) the first codeword within this equivalence class, 2) the cardinality of this equivalence class. Thus, the lexicographic order permits a compact representation of the multiplexed code table. However, note that the method is not restricted to this order.

3.3.2 Conversion of the lower priority bitstream

It appears from the above that the design of multiplexed codes relies on the choice of the partition $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_{|\mathcal{A}|}\}$. The encoding then proceeds with the conversion of the lower priority bitstream into a flow of variables taking their values in the different sets of $|\mathcal{C}_i|$-valued variables. In order to be multiplexed with symbols of $\mathbf{S}_H$, the lower priority bitstream $\mathbf{b}$ must be mapped into a sequence of $|\mathcal{C}_i|$-valued variables. Let us first consider the mapping of the realization $\mathbf{s}_H = s_1 \dots s_t \dots s_{K_H}$ into the sequence $n_1 \dots n_t \dots n_{K_H}$. The sequence of n_t-valued variables can also be seen as a unique Λ-valued variable γ, where $\Lambda = \prod_{t=1}^{K_H} n_t$. The variable γ hence verifies $0 \leq \gamma < \Lambda$. The quantity Λ denotes the number of different multiplexed sequences of codewords that can be used as a coded representation of the sequence $\mathbf{s}_H$. One of these sequences can be used as a multiplexed coded description of the two sequences $\mathbf{s}_H$ and $\mathbf{s}_L$ [1].

[1] In fact, only a part of $\mathbf{s}_L$, whose length depends on the multiplexing capacity of the sequence $\mathbf{s_H}$

The sequence of multiplexed codewords to be transmitted will then depend on the bitstream representation $\mathbf{b}$ of the source $\mathbf{S}_L$.

The maximum amount of information that can be stored in the Λ-valued variable γ is theoretically $\log_2(\Lambda)$ bits. However, since this variable is used to store bits of the bitstream $\mathbf{b}$, only $K'_B = \lfloor \log_2(\Lambda) \rfloor$ bits can be stored, leading to an overhead equal to $\frac{\log_2(\Lambda)}{K'_B}$. The last[2] K'_B bits of $\mathbf{b}$ are then seen as the binary representation of the integer γ comprised between 0 and $2^{K'_B} - 1$, that can be expressed as

$$\gamma = \sum_{r=1}^{K'_B} b_{(r+K_B-K'_B)}\ 2^{r-1}. \tag{3.2}$$

Note again that the variable γ is an index of the set of sequences that represent $\mathbf{s}_H$. The variable γ must then be expanded into a sequence of pairs (n_t, q_t) that will provide entries in the multiplexed codes table. Let us recall that q_t denotes the index of a codeword in the *equivalence class* associated with the realization s_t of a given symbol S_t. There are different ways to expand the variable γ into the sequence of n_t-valued variables (q_t), where $t = 1, \ldots, K_H$. One possible method is to use the recursive Euclidean decomposition defined below.

Définition 3.2 *The recursive Euclidean decomposition of a positive integer x by a sequence $\mathbf{y} = (y_1, y_2, \ldots, y_K)$ of positive integers is the unique sequence $\mathbf{r} = (r_1, r_2, \ldots, r_K) \in \mathbb{N}^K$ such that :*

$$\begin{cases} x = r_1 + y_1(\ldots(r_t + y_t(\cdots(r_{K-1} + y_{K-1}\, r_K)\ldots))\ldots), \\ \forall t \in \{1 \ldots K\}, 0 \leq r_t \leq y_t - 1. \end{cases} \tag{3.3}$$

It leads to the expansion of γ as $q_1 + n_1(q_2 + n_2(\ldots(q_{K_H-1} + n_{K_H-1}\, q_{K_H})\ldots))$. Since each codeword index q_t satisfies

$$q_t = \left\lfloor \frac{\gamma}{\prod_{t'=1}^{t-1} n_{t'}} \right\rfloor \mod\ n_t. \tag{3.4}$$

The components q_t can be calculated by Algorithm 1. In section 3.3.3, we will see that these quantities can be computed with a lower complexity.

In summary, the encoding proceeds as follows :

1. For $t = 1, \ldots, K_H$, let n_t be the *cardinality* of the *equivalence class* associated with the realization s_t of the symbol S_t.
2. The integer Λ is computed as $\Lambda = \prod_{t=1}^{K_H} n_t$. The number K'_B of bits of the lower priority bitstream $\mathbf{b}$ that can be jointly encoded with the sequence $\mathbf{S}_H$ is given by $\lfloor \log_2(\Lambda) \rfloor$.

[2] Any other subset of K'_B bits of $\mathbf{b}$ can be used

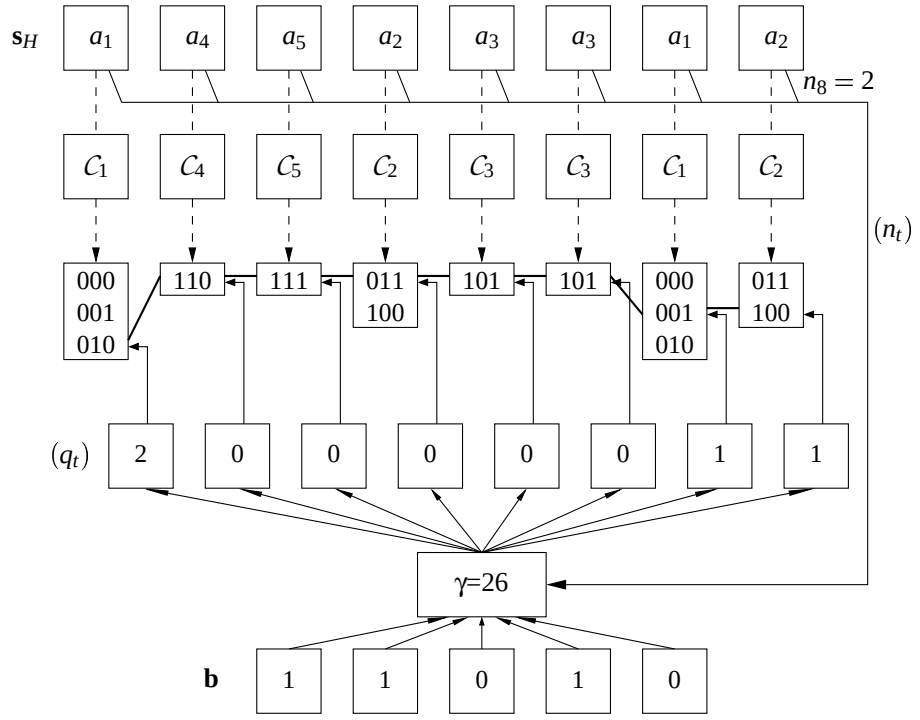


FIG. 3.1 – Link between the source $\mathbf{S}_H$, the low priority source formatted as a state flow $\mathbf{q}$ of $|\mathcal{C}_i|$-valued variables, and multiplexed codewords. Here, the bold path denotes the chosen codewords.

3. The K'_B last bits of the low priority bitstream $\mathbf{b}$ are converted into the sequence of pairs (n_t, q_t), $t = 1 \dots K_H$, using Algorithm 1 of the Euclidean decomposition of (3.3).
4. The sequence of pairs (s_t, q_t) provides the entries in the table of multiplexed codes, hence allows to select the sequence of c-bits fixed length codewords to be transmitted on the channel.
5. If $K'_B \leq K_B$, the $K_B - K'_B$ first bits of the bitstream $\mathbf{b}$ are then concatenated to the sequence of multiplexed codewords. Otherwise, the remaining symbols of $\mathbf{S}_H$ are not multiplexed but sent using other codes on the channel, such as FLCs or Huffman codes.

As all steps are reversible, the decoding proceeds in the reverse order. A schematic diagram of the encoding process, for the particular example proposed hereafter, is depicted in Fig. 3.1.

Example 3.2: Let us consider the source alphabet described in section 3.3 and the multiplexed code given in Table 3.1. Considering the sequence of symbols $\mathbf{S}_H$ given in Table 3.2, the algorithm proceeds first with the derivation of the sequence of n_t va-

Algorithm 1 Conversion of the integer γ into the sequence $\mathbf{q}$

for $t = 1$ to K_H **do**
  $q_t = \gamma' \mod n_t$
  $\gamma' = \frac{\gamma' - q_t}{n_t}$
end for

t	s_t	n_t	γ	q_t	codeword
1	a_1	3	26	2	001
2	a_4	1	8	0	000
3	a_5	1	8	0	111
4	a_2	2	8	0	011
5	a_3	1	4	0	110
6	a_3	1	4	0	111
7	a_1	3	4	1	000
8	a_2	2	1	1	001

TAB. 3.2 – Euclidean decomposition of the variable γ and corresponding multiplexed codewords.

riables as shown in Table 3.2. This leads to $\Lambda = \log_2(36) \approx 2^{5.17}$. *Therefore, the last* 5 *bits of the bitstream* $\mathbf{b} = 11010$ *are used to process the variable* $\gamma = \sum_{r=1}^{5} b_r \, 2^{r-1} = 26$. *The Euclidean decomposition of the variable* γ *according to (1) leads to the sequence of* q_t *variables (indexes of the codewords in the classes of equivalence associated with the sequence of symbols) given in Table 3.2. The sequence of pairs* (s_t, q_t) *provides directly the entries in the table of* c*-bits fixed length codewords, i.e. in the table of multiplexed codes.*

3.3.3 Hierarchical decomposition of γ

The complexity order of the Euclidean decomposition of the global variable γ is $O(K_H^2)$ if Algorithm 1 is used. This overall complexity can be reduced by considering the following hierarchical decomposition of the variable γ. Note that this reduction in terms of complexity does not induce any overhead or redundancy. The resulting states q_t are identical to the ones obtained by Algorithm 1. For sake of clarity, let us assume that $K_H = 2^l$, with $l \in \mathbb{N}$. The approach can be easily extended to any length. Let $n_{t_1}^{t_2}$ denote the product $\prod_{t=t_1}^{t_2} n_t$. Similarly, $q_{t_1}^{t_2}$ is defined as

$$q_{t_1}^{t_2} \triangleq q_{t_1} + n_{t_1}(q_{t_1+1} + \ldots (q_{t_2-1} + n_{t_2-1} q_{t_2}) \ldots). \tag{3.5}$$

Algorithm 2 Conversion of the integer γ into the sequence of states $\mathbf{q}$ using the hierarchical algorithm (for sequences such that $\exists\, l / K_H = 2^l$).

for $j = 1$ to l **do** {Processing of values $n_{t_1}^{t_2}$ involved in further processing}
  for $i = 1$ to 2^{l-j} **do**
    $t_1 = (i-1)\, 2^j + 1$
    $t_2 = i\, 2^j$
    $t_m = (t_1 + t_2 - 1)/2$
    $n_{t_1}^{t_2} = n_{t_1}^{t_m}\, n_{t_m+1}^{t_2}$
  end for
end for

for $j = l$ to 1 by -1 **do** {Processing of values q_{t1}^{t2}}
  for $i = 1$ to 2^{l-j} **do**
    $t_1 = (i-1)\, 2^j + 1$
    $t_2 = i\, 2^j$
    $t_m = (t_1 + t_2 - 1)/2$
    $q_{t_1}^{t_m} = q_{t_1}^{t_2} \mod n_{t_1}^{t_m}$
    $q_{t_{m+1}}^{t_2} = (q_{t_1}^{t_2} - q_{t_1}^{t_m})/n_{t_1}^{t_m}$
  end for
end for

If the number of terms in (3.5) is a power of two, i.e if $t_2 - t_1 + 1 = 2^{l'}$, then the entities $n_{t_1}^{t_2}$ and $q_{t_1}^{t_2}$ are respectively decomposed as

$$n_{t_1}^{t_2} = n_{t_1}^{\frac{t_2+t_1-1}{2}}\, n_{\frac{t_2+t_1+1}{2}}^{t_2}, \tag{3.6}$$

$$q_{t_1}^{t_2} = q_{t_1}^{\frac{t_2+t_1-1}{2}} + n_{t_1}^{\frac{t_2+t_1-1}{2}}\, q_{\frac{t_2+t_1+1}{2}}^{t_2}. \tag{3.7}$$

Notice that $\gamma = q_1^{2^l}$. Hence the state γ is decomposed as

$$\gamma = q_1^{2^{l-1}} + n_1^{2^{l-1}}\, q_{2^{l-1}+1}^{2^l}. \tag{3.8}$$

Similarly, each term of (3.8) can be recursively decomposed using (3.6) and (3.7). Thus, the conversion of the integer γ into a sequence of states $\mathbf{q}$ can be done iteratively, as described in Algorithm 2. Note that each level j generates the variables required for level $j+1$. The reverse algorithm follows naturally.

3.4 Complexity of the hierarchical decomposition of the variable γ

Property 3.1: the complexity order of this algorithm is in $O((K_H)^r)$, where $r > 1$ is the complexity exponent of the algorithms used for operations of large integers.

Proof: The complexity of the hierarchical decomposition of the variable γ strongly depends on the complexity of the algorithms used for the multiplication, division and modulo operations. Karatsuba [KO63] has shown that these operations have a subquadractic complexity. For example, the multiplication can be done in $O(N^{\frac{\log 3}{\log 2}})$ with the Karatsuba algorithm. For implementation purpose, efficient algorithms are described in the GNU Multiple precision computing library [GMP]. Now, let us assume that the complexity of long integer operations are given by $C(K)$, where K is the block size of the largest variable involved. Then, processing the level j in Algorithm 2 has the complexity $2^{l-j}\, C(2^j) = K_H\, 2^{-j} C(2^j)$. Assuming that $C(K) = O(K^r)$ with $r > 1$, the overall complexity can be written as

$$K_H \sum_{j=1}^{l} 2^{-j}\, O((2^j)^r). \tag{3.9}$$

Hence, from $C(K) = O(K^r)$, we deduce that $\exists A > 0\ \exists B > 0\ /\ \forall K > 0,\ C(K) < A + B\, K^r$. This leads to the following inegality

$$K_H \sum_{j=1}^{l} 2^{-j} C(2^j) < K_H \sum_{j=1}^{l} 2^{-j} (A + B\, (2^j)^r) \tag{3.10}$$

$$< K_H A \sum_{j=1}^{l} 2^{-j} + K_H\, B \sum_{j=1}^{l} 2^{j(r-1)} \tag{3.11}$$

$$< K_H\, A + K_H\, B\, 2^{(l+1)(r-1)}. \tag{3.12}$$

Since we have

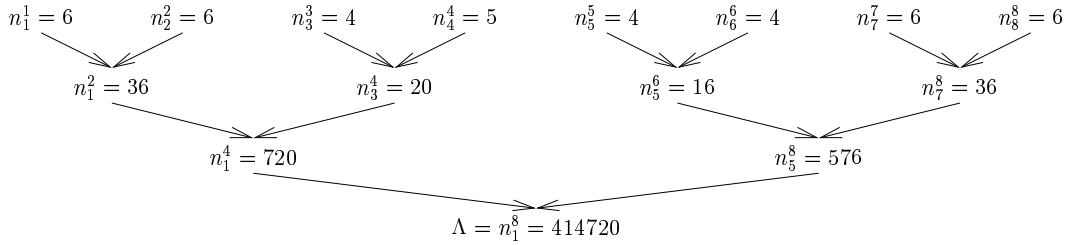
$$2^{(l+1)(r-1)} = (K_H)^{r-1}\, 2^{r-1}, \tag{3.13}$$

it appears that the complexity of processing the level l and the overall complexity are of the same order, i.e $O((K_H)^r)$. ■

Usually, the algorithm used is such that r goes from $r = \frac{\log 3}{\log 2} \approx 1.58$ to $r = \frac{\log 9}{\log 5} \approx 1.37$, depending on the length of the message [KO63] [Zur94].

Example 3.3: Let us consider the decomposition proposed in Example 3.2. This decomposition can also be processed with the hierarchical algorithm as $n_1^2 = 3$, $n_3^4 = 2$,

Processing of values $n_{t_1}^{t_2}$



Processing of values $q_{t_1}^{t_2}$

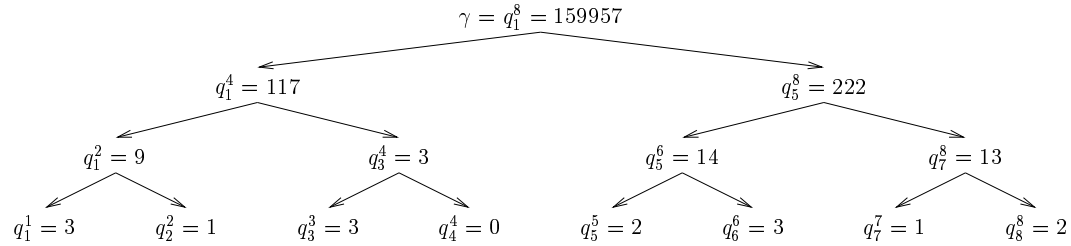


FIG. 3.2 – Hierarchical decomposition of γ.

$n_5^6 = 1$, $n_7^8 = 6$, $n_1^4 = 6$, $n_5^8 = 6$ *and* $q_1^4 = 26 \mod n_1^4 = 2$, $q_5^8 = \lfloor \frac{26}{n_1^4} \rfloor = 4$, $q_1^2 = 2$, $q_3^4 = 0$, $q_5^6 = 0$, $q_7^8 = 8$, $q_1 = 1$, $q_2 = 0$, $q_3^4 = 0$, $q_3 = 0$, $q_4 = 0$, $q_5 = 0$, $q_6 = 0$, $q_7 = 1$ *and finally* $q_8 = 1$.

The decomposition of Example 3.3 is depicted in Fig. 3.2. Note that some values $n_{t_1}^{t_2}$ processed in Fig. 3.2 are not strictly required (for example, the value $n_1^{K_H} = \Lambda$).

3.5 Compression efficiency

Each symbol of the sequence $\mathbf{S}_H$ is associated with a c-bits fixed length codeword, leading to a total length of the multiplexed flow equal to $c \times K_H$. Therefore, the compression rate is determined by the amount of $\mathbf{S}_L$ information that can be jointly coded by the sequence of multiplexed codewords. Let us consider again the realization s_t of a symbol S_t of the sequence $\mathbf{S}_H$. One can associate an n_t-valued variable to the symbol value s_t. The amount of data that can be represented by this variable is $\log_2 n_t$ bits. Since the c bits of the multiplexed codeword code both the symbol value s_t and the index q_t describing the low priority bitstream realization, the description length for the symbol s_t is theoretically $c - \log_2 n_t = -\log_2 \frac{n_t}{|\mathcal{X}|}$ bits. The amount of data, in bits,

used to code the sequence $\mathbf{s}_H$ is given by

$$-\sum_{1\leq t\leq K_H} \log_2 \frac{n_t}{|\mathcal{X}|}. \tag{3.14}$$

In order to minimize the expected description length (EDL) $\hat{h}$ of the source $\mathbf{S}_H$, one has then to choose the partitioning $\mathcal{C}$ of the set of $|\mathcal{X}|$ fixed length codewords into *equivalence classes* $\mathcal{C}_i$ of cardinality $|\mathcal{C}_i|,\ i = 1 \ldots |\mathcal{A}|$, such that

$$\begin{aligned}(|\mathcal{C}_1|, \ldots, |\mathcal{C}_i|, \ldots, |\mathcal{C}_{|\mathcal{A}|}|) &= \text{argmin}\left(-\sum_{i=1}^{|\mathcal{A}|} \mu_i \ \log_2 \frac{|\mathcal{C}_i|}{|\mathcal{X}|}\right) \\ &= \text{argmin}(\hat{h}).\end{aligned} \tag{3.15}$$

In order to have a rate close to the entropy bound of the source $\mathbf{S}_H$, the quantity $\frac{|\mathcal{C}_i|}{|\mathcal{X}|}$ should be as close as possible to μ_i. We come back in section 3.8 on how to calculate an efficient set of $|\mathcal{C}_i|$ values for a given source PMF.

Example 3.4: The entropy of the source $\mathbf{S}_H$ introduced in section 3.2 is $h = 2.122$. The partition $\mathcal{C}$ given in Table 3.1 leads to an EDL of $\mathbf{S}_H$ of $\hat{h} = 2.166$. Note that a Huffman encoder of the source would have led to an EDL equal to 2.2. The choice of c as well as of the partition $\mathcal{C}$ has an impact on the coding rate. For example, for $c = 5$, the partition into classes of respective cardinalities $(|\mathcal{C}_1|, |\mathcal{C}_2|, |\mathcal{C}_3|, |\mathcal{C}_4|, |\mathcal{C}_5|) = (13, 7, 6, 3, 3)$ produces an EDL equal to 2.124.

Property 3.2: $\forall\, \varepsilon > 0, \exists\, c \in \mathbb{N}$ such that $\hat{h} - h \leq \varepsilon$

In other words, the EDL of $\mathbf{S}_H$ can be made as close as required to the entropy, by increasing the length c of the codewords.

Proof: Let $\varepsilon > 0$. The problem consists in finding a partition $\mathcal{C}$ that verifies the property. Let c_h be the integer defined as $c_h = \lceil -\log_2(\min_{i\in\mathcal{A}}(\mu_i))\rceil$. By definition,

$$\forall\, i \in \mathcal{A},\ \mu_i \geq 2^{-c_h}. \tag{3.16}$$

For all $c \in \mathbb{N}$ such that $c \geq c_h$, we choose $|\mathcal{C}_i| = \lfloor \mu_i\, |\mathcal{X}| \rfloor$. From $\sum_{i\in\mathcal{A}} \mu_i = 1$ we get $\sum_{i\in\mathcal{A}} \lfloor \mu_i\, |\mathcal{X}| \rfloor \leq |\mathcal{X}| \sum_{i\in\mathcal{A}} \mu_i = |\mathcal{X}|$. Moreover, $\forall i,\ |\mathcal{C}_i| \geq 1$. Therefore this choice of $(|\mathcal{C}_i|)_{1\leq i\leq|\mathcal{A}|}$ is valid. By construction, $\mu_i\, |\mathcal{X}| - 1 \leq |\mathcal{C}_i| \leq \mu_i\, |\mathcal{X}|$, hence,

$$1 \leq \mu_i \frac{|\mathcal{X}|}{|\mathcal{C}_i|} \leq 1 + \frac{1}{\mu_i\, |\mathcal{X}| - 1}. \tag{3.17}$$

The difference δ_h between the EDL and the entropy is given by

$$\begin{aligned}\delta_h = \hat{h} - h &= -\sum_{i \in \mathcal{A}} \mu_i \ \log_2 \frac{|\mathcal{C}_i|}{|\mathcal{X}|} + \sum_{i \in \mathcal{A}} \mu_i \ \log_2 \mu_i \\ &= \sum_{i \in \mathcal{A}} \mu_i \ \log_2 \frac{\mu_i \, |\mathcal{X}|}{|\mathcal{C}_i|}. \end{aligned} \tag{3.18}$$

Therefore, from (3.17), it can be seen easily that δ_h verifies

$$\begin{aligned} \delta_h &\leq \textstyle\sum_{i \in \mathcal{A}} \mu_i \ \log_2(1 + \frac{1}{\mu_i \, |\mathcal{X}| - 1}) \\ \Rightarrow \ \delta_h &\leq \textstyle\sum_{i \in \mathcal{A}} \frac{\mu_i}{\mu_i \, |\mathcal{X}| - 1} = \sum_{i \in \mathcal{A}} \frac{1}{|\mathcal{X}| - \frac{1}{\mu_i}}. \end{aligned} \tag{3.19}$$

From equations (3.16) and (3.19) we get

$$\delta_h \leq \sum_{i \in \mathcal{A}} \frac{1}{|\mathcal{X}| - 2^{c_h}} = \frac{|\mathcal{A}|}{|\mathcal{X}| - 2^{c_h}}. \tag{3.20}$$

Thus, using $c = \lceil \log_2(\frac{|\mathcal{A}|}{\varepsilon} + 2^{c_h}) \rceil$ and $\forall i, \ |\mathcal{C}_i| = \lfloor \mu_i \, |\mathcal{X}| \rfloor$, the inequality $\delta_h \leq \varepsilon$ is verified.

Notice that, for any code for which the inequality $c \geq c_h$ is true, this proof provides an upper bound for the EDL. ■

3.6 Multiplexed codes based on a constrained partition $\mathcal{C}$

The computational cost of the mapping (or multiplexing) algorithm can be further reduced by considering a constrained partition of the set of fixed length codewords. The pre-formatted lower priority bitstream is in this case not seen as the representation of a unique variable γ, but as a sequence of "elementary variables". An elementary variable is defined as a variable which takes its value in a set of small dimension : it is a k-valued variable, k being small.

3.6.1 Constrained choice of partition $\mathcal{C}$

The $|\mathcal{C}_i|$-valued variables, $1 \leq i \leq |\mathcal{A}|$, are decomposed into a set of elementary variables as follows :

$$|\mathcal{C}_i| = \prod_{j=1}^{\nu_i} f_j^{\alpha_{i,j}}, \tag{3.21}$$

where f_j are prime factors. The term f_{ν_i} denotes the highest prime factor in the decomposition of $|\mathcal{C}_i|$. The term $\alpha_{i,j}$ stands for the power of the prime factor f_j in the

bit 1	bit 2	bit 3	3-valued variable 1	3-valued variable 2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	2
0	1	1	1	0
1	0	0	1	1
1	0	1	1	2
1	1	0	2	0
1	1	1	2	1
			2	2

TAB. 3.3 – Example of transformation $\mathcal{T}$ ($u_{\mathcal{T}} = 3$).

decomposition of $|\mathcal{C}_i|$. This expansion of $|\mathcal{C}_i|$-valued variables can also be represented as

$$|\mathcal{C}_i|\text{-valued variable} \rightleftarrows \begin{cases} \alpha_{i,1} \text{ binary variables} \\ \alpha_{i,2} \text{ 3-valued variables} \\ \vdots \\ \alpha_{i,\nu_i} \ f_{\nu_i}\text{-valued variables.} \end{cases} \tag{3.22}$$

In order to limit the encoding complexity, the partitioning of the set of c-bits fixed length codewords must be such that the set $(|\mathcal{C}_i|)_{1 \leq i \leq |\mathcal{A}|}$ will not result in prime decompositions into factors greater than a given value f_ν. These additional constraints on the partition $\mathcal{C}$ will induce an approximation of the PMF of the $\mathbf{S}_H$ source. This approximation leads to a slightly higher EDL. We come back on this point in Section 3.10.2.

3.6.2 Conversion of the low priority bitstream into f_j-valued variables

Consecutive segments of the low priority bitstream are seen as the binary representations of integers to be decomposed this time into a set of binary, 3-valued, ..., j-valued, ..., f_ν-valued variables. Let $\mathcal{T}$ be a transformation that takes $u_{\mathcal{T}}$ bits and produces a set of f_j-valued variables. Let $v_{\mathcal{T},j}$, $1 \leq j \leq \nu$, be the number of f_j-valued variables produced by a transformation $\mathcal{T}$. The number of possible values that can be taken by the set of f_j-valued variables resulting from the transformation $\mathcal{T}$ applied on $u_{\mathcal{T}}$ bits of the bitstream $\mathbf{b}$ is given by

$$\Lambda_{\mathcal{T}} = \prod_{j=1}^{\nu} f_j^{v_{\mathcal{T},j}}. \tag{3.23}$$

$\mathcal{T}_z$ identifier	$u_{\mathcal{T}_z}$	$v_{\mathcal{T}_z,1}$	$v_{\mathcal{T}_z,2}$	$v_{\mathcal{T}_z,3}$	$o_{\mathcal{T}_z}$
$\mathcal{T}_0$	1	1	0	0	0.0000
$\mathcal{T}_1$	15	0	8	1	0.0001
$\mathcal{T}_2$	21	0	3	7	0.0004
$\mathcal{T}_3$	19	0	12	0	0.0010
$\mathcal{T}_4$	25	0	7	6	0.0011
$\mathcal{T}_5$	24	0	2	9	0.0028
$\mathcal{T}_6$	14	0	3	4	0.0030
$\mathcal{T}_7$	18	0	7	3	0.0034
$\mathcal{T}_8$	27	0	1	11	0.0047
$\mathcal{T}_9$	17	0	2	6	0.0060
$\mathcal{T}_{10}$	30	0	0	13	0.0062
$\mathcal{T}_{11}$	20	0	1	8	0.0080
$\mathcal{T}_{12}$	11	0	7	0	0.0086
$\mathcal{T}_{13}$	23	0	0	10	0.0095
$\mathcal{T}_{14}$	6	0	1	2	0.0381
$\mathcal{T}_{15}$	3	0	2	0	0.0566
$\mathcal{T}_{16}$	2	0	0	1	0.1610
$\mathcal{T}_{z_{max}} = \mathcal{T}_{17}$	1	0	1	0	0.5850

TAB. 3.4 – Transformations used for $f_\nu = 5$

This $\Lambda_\mathcal{T}$-valued variable allows to theoretically store $\log_2(\Lambda_\mathcal{T})$ bits. Note that $\Lambda_\mathcal{T}$ may be higher than $2^{u_\mathcal{T}}$. Therefore, there is an overhead per bit given by

$$o_\mathcal{T} = \frac{\log_2(\Lambda_\mathcal{T})}{u_\mathcal{T}} - 1. \tag{3.24}$$

For example, the transformation $\mathcal{T}_{15}$ shown in table 3.3 applied on 3 bits produces two 3-valued variables, leading to a number of states increased from 8 to 9.

Table 3.4 enumerates the transformations used for $f_\nu = 5$. If $f_\nu = 3$, only the transformations that do not use 5-valued variables are valid : $\mathcal{T}_0, \mathcal{T}_3, \mathcal{T}_{12}, \mathcal{T}_{15}, \mathcal{T}_{17}$. They are sorted by increasing loss in compression efficiency. Among all the possible sets $\mathbf{v}_\mathcal{T} = (v_{\mathcal{T},1}, \dots v_{\mathcal{T},j}, \dots v_{\mathcal{T},\nu})$ explored under previous constraints, only those for which the transformations introduce an overhead lower than 1% are kept. Increasing f_ν reduces this overhead.

3.6.3 Optimal set of transformations

Let again $\mathbf{s}_h$ be the realization of a source $\mathbf{S}_H$ mapped into a sequence of n_t-valued variables. Each n_t-valued variable is decomposed into a set of f_j-valued variables, each one being used $\alpha_{t,j}$ times. Let d_j be the total number of f_j-valued va-

Algorithm 3 Choice of the transformations to be used

$z = 0$
while $sum(d_j) > 0$ **do** {while some f_j-valued variables are not computed yet}
  $g_{\mathcal{T}_z} = \lfloor \min(\frac{d_j}{v_{\mathcal{T}_z,j}}) \rfloor$ {Calculation of how many transformations $\mathcal{T}_z$ can be used}
  for $\forall j$ between 1 and ν **do** {Update the number of f_j-variables to be transformed}
    $d_j = d_j - g_{\mathcal{T}_z} \times v_{\mathcal{T}_z,j}$
    $z = z + 1$ {Try next transformation}
  end for
end while

riables involved in the expansion of the entire sequence of n_t-valued variables. Let $g_{\mathcal{T}_z}$ be the number of transformations $\mathcal{T}_z,\ z = 0, \ldots, z_{max}$ necessary to convert the low priority bitstream into the f_j-valued variables. There are several possible sets $\mathbf{g} = (g_{\mathcal{T}_1}, \ldots, g_{\mathcal{T}_z}, \ldots, g_{\mathcal{T}_{z_{max}}})$. The optimum set of transformations is the one which will minimize the total overhead, given by

$$O = \sum_{z=0}^{z_{max}} g_{\mathcal{T}_z}\ u_{\mathcal{T}_z}\ o_{\mathcal{T}_z}. \tag{3.25}$$

The choice of $\mathbf{g}$ is an optimization problem. Knowing the transformation set $(\mathbf{T}_z)_{0 \leq z \leq z_{max}}$ and $\mathbf{d} = (d_1, \ldots, d_j, \ldots, d_\nu)$, the optimal vector $\mathbf{g}$ is estimated by Algorithm 3.

3.6.4 Encoding procedure

Assuming that the low priority source s_L has been pre-encoded using efficient VLCs, the encoding of the two sequences s_H and s_L using fast multiplexed codes proceeds as follows :

1. For each prime integer f_j, $1 \leq j \leq \nu$, the number d_j of f_j-valued variables resulting from the expansion of the sequence of n_t-valued variables associated with s_H is calculated.
2. The number $g_{\mathcal{T}_z}$ of transformations $\mathcal{T}_z$ needed to convert the low priority bitstream $\mathbf{b}$ into a sequence of f_j-valued variables is searched with the procedure described in Section 3.6.3. The transformations $\mathcal{T}_z$ are then applied on consecutive segments of $u_{\mathcal{T}_z}$ bits of the bitstream $\mathbf{b}$, leading to the generation of sequences of f_j-valued variables.
3. The n_t-valued variable associated with the symbol realization s_t has to be in bijection with several elementary f_j-valued variables generated in the previous step. Then, at this point, the encoder has to choose the value q_t taken by this n_t-valued variable, such that the Euclidean decomposition of q_t leads to the given sequence of f_j-valued variables. This is done using the Euclidean multiplication.

t	1	2	3	4	5	6	7	8	
s_t	a_1	a_4	a_5	a_2	a_3	a_3	a_1	a_2	
n_t	3	1	1	2	1	1	3	2	
$\alpha_{t,1}$	0	0	0	1	0	0	0	1	$d_1 = 2$
$\alpha_{t,2}$	1	0	0	0	0	0	1	0	$d_2 = 2$

TAB. 3.5 – Calculation of the number of available f_j-valued variables.

4. The resulting pair (s_t, q_t) provides entries in the multiplexed codewords table to select the codeword to be transmitted.

The decoding algorithm is constructed in the reverse order. As a consequence, it is not necessary to receive the whole sequence of codewords to decode the information of high priority $\mathbf{s_H}$: it is directly read in the multiplexed codes tables. Moreover, random access is also possible for the flow $\mathbf{s_H}$.

Example 3.5: Let us consider the sequence of 8 *high priority symbols given in Table 3.5, with the corresponding mapping into the sequence of n_t-valued variables according to the multiplexed codes given in Table 3.1. The sequence of n_t-valued variables is decomposed into a sequence of d_1 binary and d_2 3-valued variables with respective powers $\alpha_{t,1}$ and $\alpha_{t,2}$ (cf. table 3.5). In a second step, one has to convert the low priority bitstream into the binary and* 3*-valued variables. It appears that both transformations $\mathcal{T}_0$ and $\mathcal{T}_{15}$ have to be used (see section 3.6.3)* 2 *times and* 1 *time, respectively. Thus,* 5 *bits can be multiplexed with the realization $\mathbf{s_H}$ of $\mathbf{S}_H$. The transformation $\mathcal{T}_0$ being identity, the first* 3 *bits of the bitstream* $\mathbf{b}$ *are multiplexed unchanged. For each transformation $\mathcal{T}$, the $u_{\mathcal{T}}$ input bits are seen as the binary representation of an integer $\gamma_{\mathcal{T}}$ as*

$\mathcal{T}$		*Input bits*	$\rightarrow$	$\gamma_{\mathcal{T}}$	$\rightarrow$	*binary*	3*-valued*
$\mathcal{T}_0$	:	1	$\rightarrow$	1	$\rightarrow$	1	
$\mathcal{T}_0$	:	1	$\rightarrow$	1	$\rightarrow$	1	
$\mathcal{T}_{15}$	:	010	$\rightarrow$	2	$\rightarrow$		02

and leading to the sequences 11 *of binary variables and* 02 *of* 3*-valued variables. The value q_t taken by the n_t-valued variable associated with the symbol realization s_t and to the realization of the low priority bitstream* $\mathbf{b}$ *is obtained from the sequences of binary ($j = 1$) and* 3*-valued ($j = 2$) variables.*

In the example above, the state q_t is generated using either a bit for equivalence class $\mathcal{C}_2$, either a 3-valued variable for equivalence class $\mathcal{C}_1$. The resulting pair (s_t, q_t) provides entries in the multiplexed codes table (see Table 3.1), leading to the selection of multiplexed codes given in Table 3.6.

t	1	2	3	4	5	6	7	8
s_t	a_1	a_4	a_5	a_2	a_3	a_3	a_1	a_2
binary variables				1				1
3-valued variables	0						2	
q_t	0	0	0	1			2	1
codeword	000	110	111	100	101	101	010	100

TAB. 3.6 – Construction of the sequence of multiplexed codewords.

class $\mathcal{C}_i$	codeword x	symbol a_i	$\lvert\mathcal{C}_i\rvert$	probability μ_i	$\overline{U_i}$
$\mathcal{C}_5$	000	a_5	1	0.10	$\varnothing$
$\mathcal{C}_1$	001	a_1	2	0.40	0
	010				1
$\mathcal{C}_2$	011	a_2	2	0.20	0
	100				1
$\mathcal{C}_3$	101	a_3	2	0.20	1
	110				0
$\mathcal{C}_4$	111	a_4	1	0.10	$\varnothing$

TAB. 3.7 – A binary Multiplexed codes ($c = 3$, EDL $= 2.2$).

3.7 Binary multiplexed codes

The partition of the FLC into several equivalence classes can be chosen such that the storage capacity is an integer number of bits. It means that, for each equivalence class $\mathcal{C}_i$, a bijection exists between the interval of indexes $\{0, |\mathcal{C}_i| - 1\}$ and a varying integer number of bits, hence the following definition :

Définition 3.3 *A binary multiplexed code is a multiplexed code such that*

$$\forall i \in \{1, \ldots, |\mathcal{A}|\},\ \log_2(|\mathcal{C}_i|) \in \mathbb{N}.$$

Then, to each symbol a_i of the source $\mathbf{S}_H$, one can associate an equivalence class $\mathcal{C}_i$, hence a set of fixed length codewords, which is in turn indexed by a sequence $\overline{U_i} = U_i^1 \ldots U_i^{\log_2|\mathcal{C}_i|}$ of $\log_2(|\mathcal{C}_i|)$ bits. A codeword representing jointly a symbol of the source $\mathbf{S}_H$ and a segment of the low priority bitstream $\mathbf{b}$ can thus be selected without requiring any conversion of the bitstream $\mathbf{b}$.

Example 3.6: Let again consider the source $\mathbf{s}_H = a_1a_4a_5a_2a_3a_3a_1a_2$ *and* $\mathbf{b} = 110100$ *a pre-encoded bitstream. Let us also consider the binary multiplexed codes of Table 3.7. The number of bits that can be multiplexed with each symbol realization of the se-*

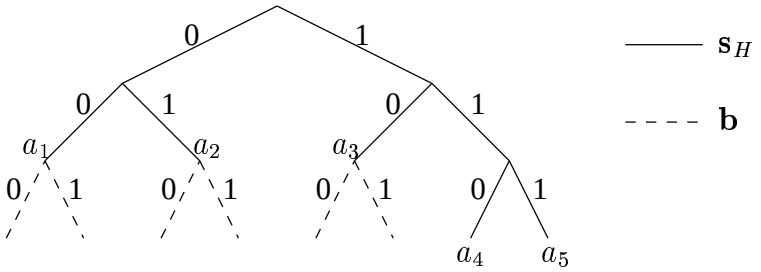


FIG. 3.3 – Codetree describing both $\mathbf{S}_H$ and $\mathbf{B}$

quence $\mathbf{S}_H$ *is processed, which leads to segment* $\mathbf{b}$ *into*

$$(\overline{u_1}, \ldots, \overline{u_t}, \ldots, \overline{u_{K_H}}) = (1, \varnothing, \varnothing, 1, 0, 1, 0, 0),$$

where Notation $\varnothing$ *indicates that the corresponding equivalence class is of cardinality 1 and is consequently not indexed by any bit. Then, for each high priority symbol index* t*, the couple* $(s_t, \overline{u_t})$ *indexes a codeword in the multiplexed table. The resulting multiplexed bitstream is* 010 111 000 100 110 101 001 011*.*

Instead of designing a FLC codebook for the high priority source $\mathbf{S}_H$, one can alternatively consider a variable length codetree. This VLC codetree can then be completed to form a binary FLC codetree (see Fig. 3.3). The symbols of the alphabet of the source $\mathbf{S}_H$ will then be associated either to leaves of the FLC or to intermediate nodes. When they correspond to a node, all the corresponding leaves will constitute the equivalence class associated with the given symbol of $\mathbf{S}_H$. The cardinalities of the equivalence classes are the integer powers of 2. Hence, one can create a multiplexed code such that codewords of the same equivalence class have the same prefix. The suffix $\overline{U_i}$ of the codewords can then be used to "store" the bits of the bitstream $\mathbf{b}$ (see Fig. 3.3).

Définition 3.4 *A Binary multiplexed code derived from a VLC is a multiplexed code constructed such that (1) every prefix of a codeword is the representation of a symbol in a VLC tree, (2) every suffix corresponds to the multiplexed data of the low priority bitstream.*

Example 3.7: considering the same sequences as in example 3.6, the VLC-based multiplexed code depicted in Fig. 3.3 leads to the multiplexed bitstream below.

s_t	a_1	a_4	a_5	a_2	a_3	a_3	a_1	a_2
prefix	00.	110	111	01.	10.	10.	00.	01.
$\overline{u_t}$	..1	...	...	..1	..0	..1	..0	..0
codeword	001	110	111	010	100	101	000	010

The compression efficiency for the source $\mathbf{S}_H$ is obviously given by the efficiency of the VLC considered. Similarly, since the low priority source before multiplexing has been pre-encoded, the corresponding compression efficiency depends on the VLC code considered for this source. The low priority source (e.g high frequencies of a wavelet decomposition) can be pre-encoded with an arithmetic coder.

3.8 Probability distribution function approximation

As already mentioned above, the EDL $\hat{h}$ of the source $\mathbf{S}_H$ is function of the sizes $|\mathcal{C}_i|$ of the equivalence classes $\mathcal{C}_i, i = 1 \dots |\mathcal{A}|$. This section describes an algorithm that aims at approximating, under constraints, the PMF of the source $\mathbf{S}_H$. The probability distribution function estimator $\hat{\boldsymbol{\mu}}$ is defined by

$$\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \dots, \hat{\mu}_i, \dots, \hat{\mu}_{|\mathcal{A}|}) = \left(\frac{|\mathcal{C}_1|}{|\mathcal{X}|}, \dots, \frac{|\mathcal{C}_i|}{|\mathcal{X}|}, \dots, \frac{|\mathcal{C}_{|\mathcal{A}|}|}{|\mathcal{X}|} \right), \tag{3.26}$$

where $\sum_{i=1}^{|\mathcal{A}|} |\mathcal{C}_i| \leq |\mathcal{X}|$. Let $\boldsymbol{\eta} = \{\eta_1 = 1, \dots, \eta_k, \dots\}$ be the set of possible values $|\mathcal{C}_i|$. A higher number of valid values for $|\mathcal{C}_i|$ leads indeed to a better approximation of the PMF of the source $\mathbf{S}_H$. The probability μ_i of a symbol cannot be lower than $\frac{1}{|\mathcal{X}|}$. The alphabet $\mathcal{A}$ is partitioned into two subsets $\mathcal{A}_m$ and $\mathcal{A}_M$, defined respectively by

$$a_i \in \mathcal{A}_m \text{ iff } \mu_i < \frac{1}{|\mathcal{X}|} \text{ and } a_i \in \mathcal{A}_M \text{ iff } \mu_i \geq \frac{1}{|\mathcal{X}|}. \tag{3.27}$$

Let $\tilde{\boldsymbol{\mu}}$ be the probability distribution function of the set of symbols $a_i \in \mathcal{A}_M$. Each is given by

$$\tilde{\mu}_i = \frac{\mu_i}{\sum_{a_i \in \mathcal{A}_M} \mu_i} \tag{3.28}$$

The algorithm aiming at the best partition of the set of codewords proceeds as follows :

1. For each $a_i \in \mathcal{A}_m$, $|\mathcal{C}_i|$ is set to 1.
2. The probability density function $\tilde{\boldsymbol{\mu}}$ of symbols belonging to $\mathcal{A}_M$ is calculated. Since $|\mathcal{A}_m|$ codewords have already been affected to equivalence classes in the first step, there is still $|\mathcal{X}| - |\mathcal{A}_m|$ codewords to be assigned to equivalence classes. The expression (3.15) can then be written as
   $(|\mathcal{C}_1|, \dots, |\mathcal{C}_i|, \dots, |\mathcal{C}_{|\mathcal{A}|}|) =$

$$\text{argmin} \left(\log_2(|\mathcal{X}|) \sum_{a_i \in \mathcal{A}_m} \mu_i - \sum_{a_i \in \mathcal{A}_M} \mu_i \ \log_2(\frac{|\mathcal{C}_i|}{|\mathcal{X}|}) \right). \tag{3.29}$$

Since the first part of expression (3.29) is a constant and $|\mathcal{C}_i| = 1$ when $a_i \in \mathcal{A}_m$, the optimization is performed for symbols of $\mathcal{A}_M$ only, and is expressed as

$$\begin{aligned}(|\mathcal{C}_i|)_{i\in\mathcal{A}_M} &= \text{argmin}\left(-\textstyle\sum_{a_i\in\mathcal{A}_M}\mu_i\ \log_2(\frac{|\mathcal{C}_i|}{|\mathcal{X}|})\right)\\ &= \text{argmin}\left(-\textstyle\sum_{a_i\in\mathcal{A}_M}\frac{\mu_i}{\sum_{i\in\mathcal{A}_M}\mu_i}\ (\log_2(\frac{|\mathcal{C}_i|}{|\mathcal{X}|-|\mathcal{A}_m|})\right.\\ &\qquad\left.+\log_2(\frac{|\mathcal{X}|-|\mathcal{A}_m|}{|\mathcal{X}|}))\right)\\ &= \text{argmin}\left(-\textstyle\sum_{a_i\in\mathcal{A}_M}\tilde{\mu}_i\ \log_2(\frac{|\mathcal{C}_i|}{|\mathcal{X}|-|\mathcal{A}_m|})\right.\\ &\qquad\left.-\textstyle\sum_{a_i\in\mathcal{A}_M}\tilde{\mu}_i\ \log_2(\frac{|\mathcal{X}|-|\mathcal{A}_m|}{|\mathcal{X}|})\right)\\ &= \text{argmin}\left(-\textstyle\sum_{a_i\in\mathcal{A}_M}\tilde{\mu}_i\ (\log_2(\frac{|\mathcal{C}_i|}{|\mathcal{X}|-|\mathcal{A}_m|})\right).\end{aligned} \tag{3.30}$$

As $\sum_{a_i\in\mathcal{A}_M}\tilde{\mu}_i = 1$, expression (3.30) can be seen as the expression of the EDL when the index i is constrained to be such that $a_i \in \mathcal{A}_M$. Consequently, assuming that $\frac{|\mathcal{C}_i|}{|\mathcal{X}|-|\mathcal{A}_m|}$ can take any value of interval $[0, 1]$, the optimum is given by :

$$\forall a_i \in \mathcal{A}_M,\ \frac{|\mathcal{C}_i|}{|\mathcal{X}| - |\mathcal{A}_m|} = \tilde{\mu}_i. \tag{3.31}$$

In the following, the index i is supposed to be chosen such that $a_i \in \mathcal{A}_M$.

3. For each $a_i \in \mathcal{A}_M$, $|\mathcal{C}_i|$ is set to the highest value in $\boldsymbol{\eta}$ such that $|\mathcal{C}_i| \leq \tilde{\mu}_i\,(|\mathcal{X}| - |\mathcal{A}_m|)$. However, at this point, some cardinalities may be equal to zero. In that case, the corresponding symbols are assumed to belong to $\mathcal{A}_m$ instead of $\mathcal{A}_M$ and the algorithm goes back to step 1. The remaining number of codewords to be assigned to equivalence classes is then given by $\sigma = |\mathcal{X}| - \sum_{i=1}^{|\mathcal{A}|}|\mathcal{C}_i| \geq 0$.
4. As a consequence, it is possible to increase some values of $|\mathcal{C}_i|$. For each index i between 1 and $|\mathcal{A}|$, if the value η_{k+1} is used instead of η_k, the EDL $\hat{h}$ decreases of a positive value $\mu_i\ \log_2(\frac{\eta_{k+1}}{\eta_k})$. The decreasing EDL is induced by the assignment of $\eta_{k+1}-\eta_k$ pairs $(a_i, |\mathcal{C}_i|)$ to classes of equivalence and, in average per codeword, is given by

$$\gamma_i = \frac{\mu_i\ \log_2(\frac{\eta_{k+1}}{\eta_k})}{\eta_{k+1} - \eta_k}. \tag{3.32}$$

These pairs are then sorted by decreasing values of γ_i. The way to proceed is to choose the pair $(a_i, |\mathcal{C}_i|)$ with the highest associated value γ_i. For a given variable $|\mathcal{C}_i|$ set in the previous steps, it is possible to re-set it to η_{k+1} instead of η_k only if $\eta_{k+1} - \eta_k \leq \sigma$, i.e. if there is a sufficient number of codewords still to be assigned. If it is not the case, the pair $(a_i, |\mathcal{C}_i|)$ is ignored in the next iteration of the algorithm. The procedure continues with the treatment of the pair with the next value γ_i. If $\eta_{k+1} - \eta_k \leq \sigma$, then $|\mathcal{C}_i| = \eta_{k+1}$, the value σ is set to $\sigma - \eta_{k+1} + \eta_k$, and the variable γ_i is updated for this symbol.

3.9 Error handling on a binary symmetric channel

In this section, we analyze the impact of channel errors on the distortion of the source $\mathbf{S}_H$. The part of the flow $\mathbf{b}$ that is not multiplexed is supposed to be lost if an error occurs on the channel. We consider a BSC with a bit error rate p. The PMF $\boldsymbol{\mu}$ of $\mathbf{S}_H$ being known, it is possible to calculate the reconstruction accuracy in terms of SER or of MSE. Since each state q_t calculated from $\mathbf{b}$ can be seen as a uniform random variable, a symbol a_i of $\mathcal{A}$ has the same probability to be encoded with any codeword of its equivalence class $\mathcal{C}_i$. Hence, for a given codeword $x \in \mathcal{C}_i$,

$$P(x) = \frac{\mu_i}{|\mathcal{C}_i|}. \tag{3.33}$$

The probability to receive the symbol y, if x has been emitted, is function of the Hamming distance $H(x, y)$ between the codewords x and y, and is given by $P(y/x) = p^{H(x,y)}\,(1-p)^{c-H(x,y)}$. The mean distortion induced by the channel noise is then given by

$$\mathbb{E}(\Delta_{\mathbf{S}_H}) = \sum_{x\in\mathcal{X}} P(x) \sum_{y\in\mathcal{X}} P(y/x)\ \Delta(x, y), \tag{3.34}$$

where $\Delta(x, y)$ denotes the distortion of the source $\mathbf{S}_H$ induced by the reconstruction of x with y. Let respectively a_i and $a_{i'}$ be the symbols decoded from x and y and $\Delta(a_i, a_{i'})$ the corresponding distortion measure. E.g., $\Delta(a_i, a_{i'})$ may be, respectively, the Kronecker operator or the quadratic error, depending on the error-resilience measure (respectively the SER or the MSE). This leads to

$$\mathbb{E}(\Delta_{\mathbf{S}_H}) = \sum_{a_i\in\mathcal{A}} \mu_i \sum_{a_{i'}\in\mathcal{A}} \Delta(a_i, a_{i'}) \underbrace{\frac{1}{|\mathcal{C}_i|} \sum_{x\in\mathcal{C}_i} \sum_{y\in\mathcal{C}_{i'}} P(y/x)}_{P(a_{i'}/a_i)}, \tag{3.35}$$

where $P(a_{i'}/a_i)$ is the probability to decode the symbol $a_{i'}$ if a_i has been transmitted. Therefore, the choice of the partition $\mathcal{C}$ has a major impact on the final distortion. However, the number of partitions for a given set of $|\mathcal{C}_i|$-values is very high, and is given by $\frac{|\mathcal{X}|!}{\prod_{i=1}^{|\mathcal{A}|} |\mathcal{C}_i|!}$. For example, the number of partitions such that $|\mathcal{C}_1| = 3, |\mathcal{C}_2| = 2, |\mathcal{C}_3| = 1, |\mathcal{C}_4| = 1, |\mathcal{C}_5| = 1$, is 3 360.

Example 3.8: Let us consider again the example of the source alphabet given in section 3.3 and the corresponding multiplexed code of codeword length $c = 6$. Let us define the alphabet values as $a_1 = 1$, $a_2 = 2$, $a_2 = 3$, $a_3 = 4$, $a_4 = 5$. We consider the two following distortion measures : the SER (in that case $\Delta(a_i, a_{i'}) = 1$ if $a_i = a_{i'}$, 0 otherwise) and the quadratic error ($\Delta(a_i, a_{i'}) = (a_i - a_{i'})^2$). For a BSC of bit error rate (BER) equal to 0.01, the summation in (3.35) leads to a SER of 0.0279 and an MSE of 0.0814 for a multiplexed code when a lexicographic IA is used. If the IA is optimized

using a simulated annealing algorithm (e.g, [Far90]), then the distortion is 0.0243 *for the SER. However, we did not observe any improvement for the MSE criteria using this method (for this source).*

For multiplexed codes derived from a VLC, the analysis of the SER simplifies as follows. One can remark that a given symbol a_i is in error if and only if the prefix that identifies the equivalence class is in error. Hence, the probability that the symbol a_i is in error is given by $p^{c-\log_2(|\mathcal{C}_i|)}$. This leads to the following expression of the SER :

$$\text{SER}_{\mathbf{S}_H} = 1 - \sum_{a_i \in \mathcal{A}} \mu_i \, (1-p)^{c-\log_2(|\mathcal{C}_i|)}. \tag{3.36}$$

For low BER, the approximation $(1-p)^n = 1 - n\,p + O(p^2)$ leads to

$$\text{SER}_{\mathbf{S}_H} = p \sum_{a_i \in \mathcal{A}} \mu_i \, (c - \log_2(|\mathcal{C}_i)|) + O(p^2), \tag{3.37}$$

where we recognize the expression of the EDL of the VLC code from which the multiplexed code has been derived : SER $\approx p \times$ EDL. This SER is lower than for FLCs ($p \times EDL \leq p \times c$), but the error-resilience is not uniformly better : symbols with higher probabilities of occurrence have a lower probability to be erroneous whereas symbols with lower probabilities have a higher probability to be erroneous.

Note on the error-resilience of the low priority bitstream : in the general case and from equation (3.3), it appears that any error occurring on a multiplexed codeword at symbol clock t engenders, at least, an error on the quantity n_t or on the state q_t, or on possibly both. Consequently, the remaining states $q_{t+1} \dots q_{K_H}$ are corrupted. Thus, the conversion of $\mathbf{S}_L$ into a sequence of state is a step that is sensitive to bit errors. As a first approximation, the expectation of the position of the first impaired bit on the BSC is given by $(c - \text{EDL}_{\mathbf{S}_H})/(c \times p)$ (instead of $1/p$ if $\mathbf{S}_L$ is not multiplexed).

In binary multiplexed codes, the bits of $\mathbf{S}_L$ are directly indexed by codewords. It follows that either a bit error may lead to drop or to insert some bits, either the number of decoded bits is correct but these bits are erroneous. Consequently, the bitstream is not fully desynchronized in terms of Levenshtein distance. This fact has been validated by simulations results (see section 3.11).

3.10 Discussion and Parameters choice

3.10.1 Redundancy in terms of the ratio $\mathbf{S}_H$ versus $\mathbf{S}_H$+$\mathbf{S}_L$

We have seen above that c must be high enough so that one can find a partition that will closely approximate the PMF of the source $\mathbf{S}_H$. On the other hand, a high value for the parameter c will result in a decreased rate of synchronous (high priority)

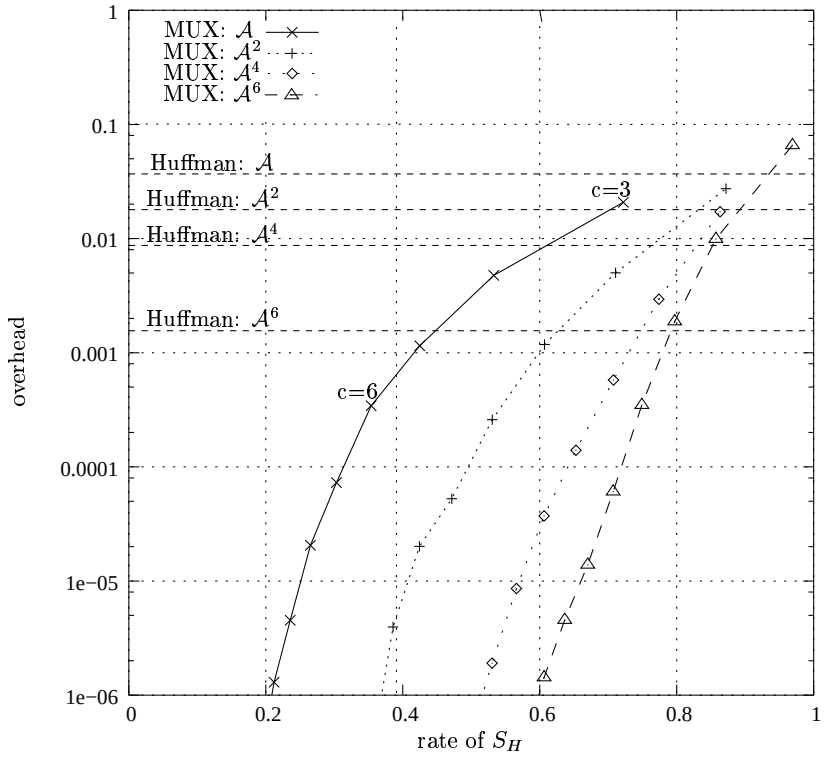


FIG. 3.4 – Compression efficiency *vs* Ratio $\frac{\mathbf{S}_H}{\mathbf{S}_H+\mathbf{S}_L}$. The source is the one given in Example 3.1. Multiplexed codes designed for the product alphabets $\{a_1, a_2, a_3, a_4, a_5\}^n$ with $n = 1, 2, 4, 6$ have been considered. Compression efficiencies of the corresponding generalized Huffman codes are also depicted.

data. Thus a compromise has to be found between reaching the entropy (compression efficiency) and the rate of synchronous data in the multiplexed flow (error-resilience). Fig. 3.4 shows the overhead (or redundancy) in terms of the ratio between the source $\mathbf{S}_H$ EDL and the total amount of transmitted bits ($\mathbf{S}_H$+$\mathbf{S}_L$). It can be seen that, if we consider multiplexed codes for joint encoding of motion and texture information in a video stream, for which realistic ratios are around 15%, the redundancy of the code is negligible (around 10^{-6}).

Fig. 3.4 shows the performance of multiplexed codes versus Huffman codes when both codes are designed for product alphabets. It can be observed that when considering a ratio of S_H versus the total number of symbols of 60%, the overhead remains rather low (10^{-3}). To achieve the same overhead with a Huffman code, one has to take the symbols by groups of 6 symbols (alphabet $\mathcal{A}^6$). Fig. 3.4 also shows that using product alphabets (here $\mathcal{A}^2$, $\mathcal{A}^4$ and $\mathcal{A}^6$) permits a better trade-off between compression efficiency and the ratio of the high priority source versus the overall number of

symbols, however at the cost of larger codetrees or tables. Note that using product alphabets preserves the strict-sense synchronization of the multiplexed codes. However, this hard synchronization is in this case guaranteed every n symbols, where n is the power of the alphabet.

3.10.2 PMF approximation inherent to codes with constrained partitions

Let us consider the codes based on a constrained partition presented in Section 3.6.1. The ability to closely approximate the PMF of the source $\mathbf{S}_H$ (hence the closeness to entropy) depends also on the parameter ν. The choice of the parameters c and ν has hence a strong impact on the performance, in terms of compression and error-resilience, of the codes. Let h be the entropy per symbol of the source and $\hat{h}$ the mean codeword length produced by the algorithm. The overhead rate is given by $\frac{\hat{h}-h}{h}$. As c bits are always used to represent a symbol of the source $\mathbf{S}_H$, the rate of high priority data in the final multiplexed c-bits codewords stream is given by $\frac{h}{c}$. Fig. 3.5 shows the influence of the parameters c and ν on the rate $\frac{\hat{h}-h}{h}$. For a source of cardinality $|\mathcal{A}| = 256$, the value $f_\nu = 5$ leads to a small overhead when the codeword length is $c = 14$. Notice that the number of available transformations and the number of flows of f_j-valued variables increases with ν. Therefore, the computational cost of the algorithm increases with ν.

The problem of finding good cardinalities for binary multiplexed codes is equivalent to the problem of finding good VLC binary codetrees. Thus, choosing the codewords length is equivalent to minimizing (3.15) by assigning the cardinalities $|\mathcal{C}_i|$, under the constraints $\log_2(|\mathcal{C}_i|) \in \mathbb{N}$ and $\sum_{a_i \in \mathcal{A}} |\mathcal{C}_i| = 2^c$. It is well known that the algorithm that will provide the best approximation of the source $\mathbf{S}_H$ PMF is the Huffman algorithm [Huf52]. However, in order to meet the requirements in terms of the ratio of synchronous data, one may limit the length of the longest codeword [Voo74], which amounts to limit the parameter c. For this purpose, the algorithm proposed in [LH90] may be used.

3.10.3 Elements of encoding and decoding complexity

The conversion of the low priority source into a sequence of states is the step that has the highest computing cost. For binary multiplexed codes, we have seen in section 3.7 that this conversion is straightforward. In that case, the subsequent computing costs required for encoding and decoding are very similar to the ones involved in encoding and decoding VLCs. The encoding and decoding of multiplexed codes based on constrained partitions (see Section 3.6) has a higher computing cost but the order of complexity remains the same : the number of operations involved in the conversion of the low priority source is linear as the length of the sequence to encode increases. In section 3.3.3, we have shown that the complexity of the general case is sub-quadratic, but above linear complexity. The exact complexity depends on the algorithms used for large integer multiplications and divisions.

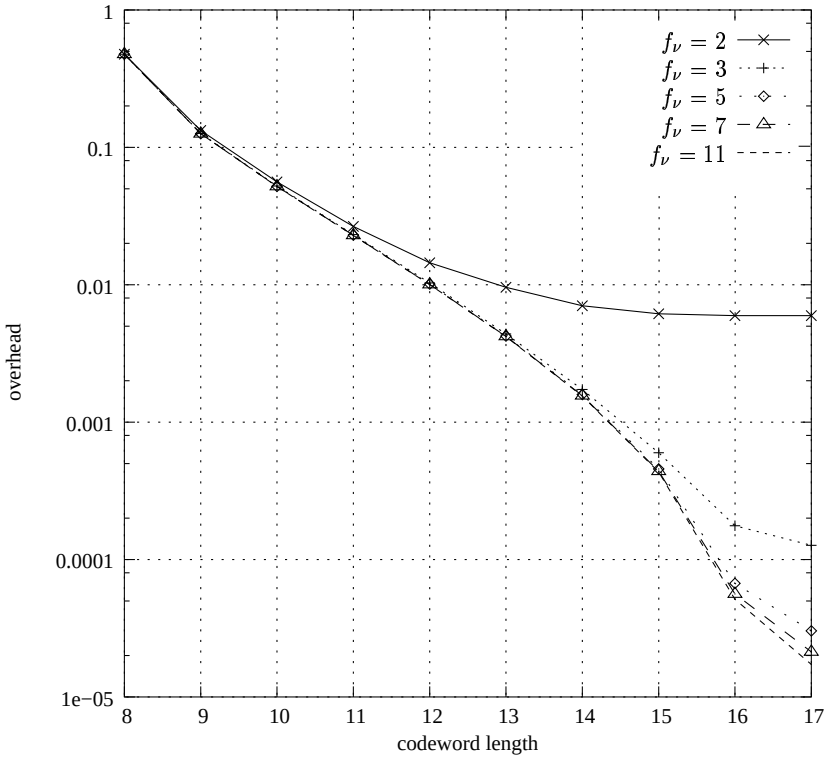


FIG. 3.5 – Closeness to entropy of fast-computable Multiplexed codes (Gaussian source).

Fig. 3.6 depicts quantitative results of encoding/decoding complexity using an implementation of Algorithm 2 based on the GNU Multiple Precision Arithmetic Library [GMP]. The computing cost, expressed in seconds, is compared with a typical implementation of Huffman codes. It is shown that the complexity of multiplexed codes, as the sequence length increases, remains tractable for most applications. Multiplexed codes complexity has also been compared against the complexity obtained with soft decoding of VLCs based on a bit/symbol trellis [BH00d] and on the BCJR algorithm [BCJR74]. Fig. 3.6 shows that this solution has a rather higher complexity than multiplexed codes. This results from the fact that the corresponding trellis has a number of states which is quadratic as the length of the sequence increases, due to the necessity to preserve the symbol clock in the state model. Note that many authors have proposed methods to decrease the complexity of VLCs with a termination length constraint, using pruning techniques or suboptimal trellis, e.g. [Wei03], but then the estimation is not optimal.

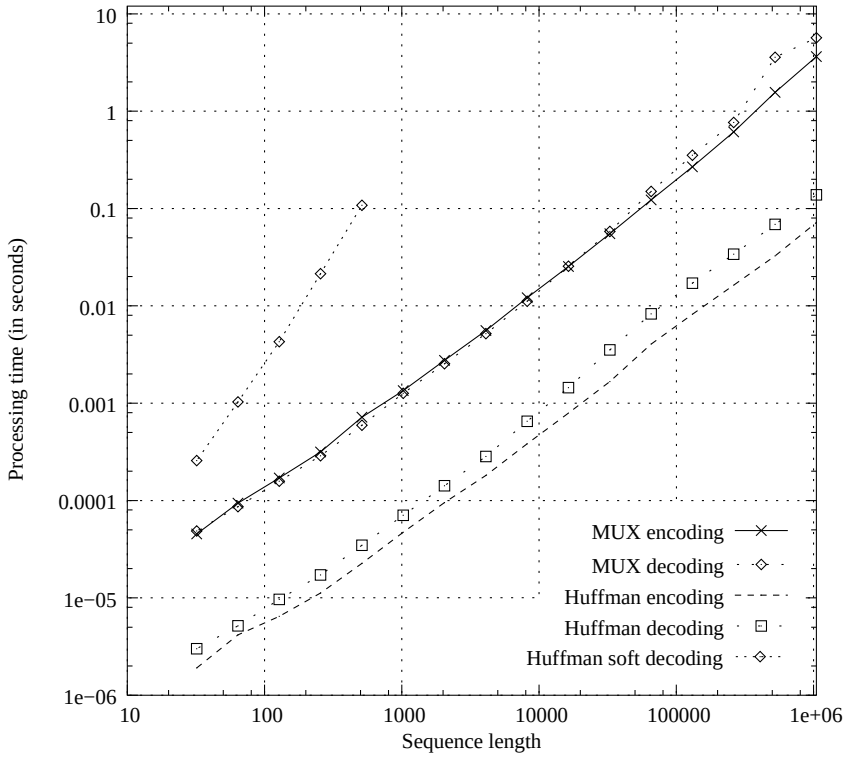


FIG. 3.6 – Encoding and decoding processing durations of Multiplexed codes (c=3) based on Algorithm 2. Source of Example 3.1 has still been considered. Corresponding durations are provided for encoding and decoding (hard and soft) of Huffman codes.

3.11 Simulation results

The multiplexed code are most beneficial when applied to sources of different levels of priority. They intrinsically support unequal error protection of the corresponding sources. This is going to be illustrated below with a simple image coding system.

However, these codes can also be applied in the case, there is a unique source. They allow one to guarantee hard synchronization for at least part of the sequence of symbols to be transmitted. To illustrate the advantage in such a situation, the performance of the multiplexed codes referred to as MUX codes in the sequel and on the figures) is first assessed in terms of SER (Hamming distance) and Levenshtein distance by considering the simple theoretical source given in Example 3.1 [MR85]. The performance of the MUX code is compared to the performance achieved with the VLC $\{01, 00, 11, 100, 101\}$ referred to as $C5$ in [ZZ02]. We have chosen to use this code as a basis for comparisons since, among the 16 optimal codes described in [ZZ02], this is

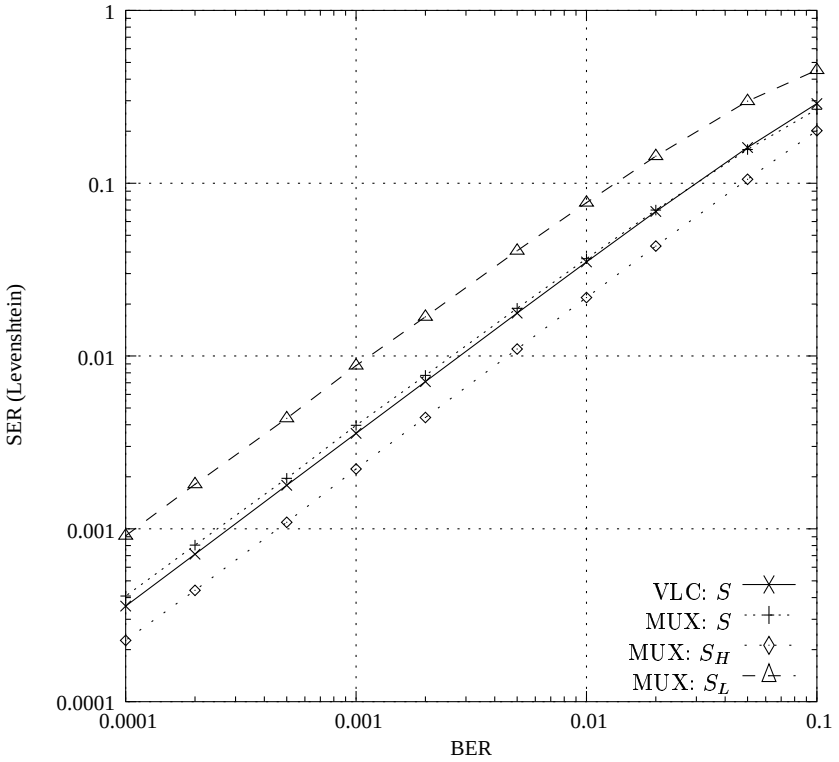


FIG. 3.7 – Performance in terms of normalized Levenshtein distance of multiplexed codes *vs* Huffman codes.

the code which leads to the best performance in terms of Levenshtein distance. For the experiments reported for the theoretical source, we have considered sequences of 1000 symbols. The sequence of 1000 symbols is divided into two segments. The first and second segments are assumed to be the sources of high and low priority respectively. The second segment of the sequence of symbols is thus coded with this code simply referred to as VLC on the figures. The sequence is divided in such a way that the multiplexing capacity of the high priority source is greater or equal to the length of the bitstream representation of the low priority source. The multiplexed code used in this first experiment is the lexicographic binary multiplexed code such that $|\mathcal{C}_1|$ = 4, $|\mathcal{C}_2|$ = 1, $|\mathcal{C}_3|$ = 1, $|\mathcal{C}_4| = 1$ and $|\mathcal{C}_5|$ = 1. This code is derived from the VLC code $\{0, 100, 101, 110, 111\}$.

This code leads to a ratio between the high priority symbols to the overall number of symbols equal to 0.72. The EDL measured is close to the theoretical value of 2.2 with both the MUX and the VLC $(C5)$ codes. The results have been averaged over 20000 simulations. For both codes, we have respectively measured the overall performance in terms of normalized Levenshtein distance and SER. Fig. 3.7 and Fig. 3.8 show the performance in terms of normalized Levenshtein distance and SER obtained on the

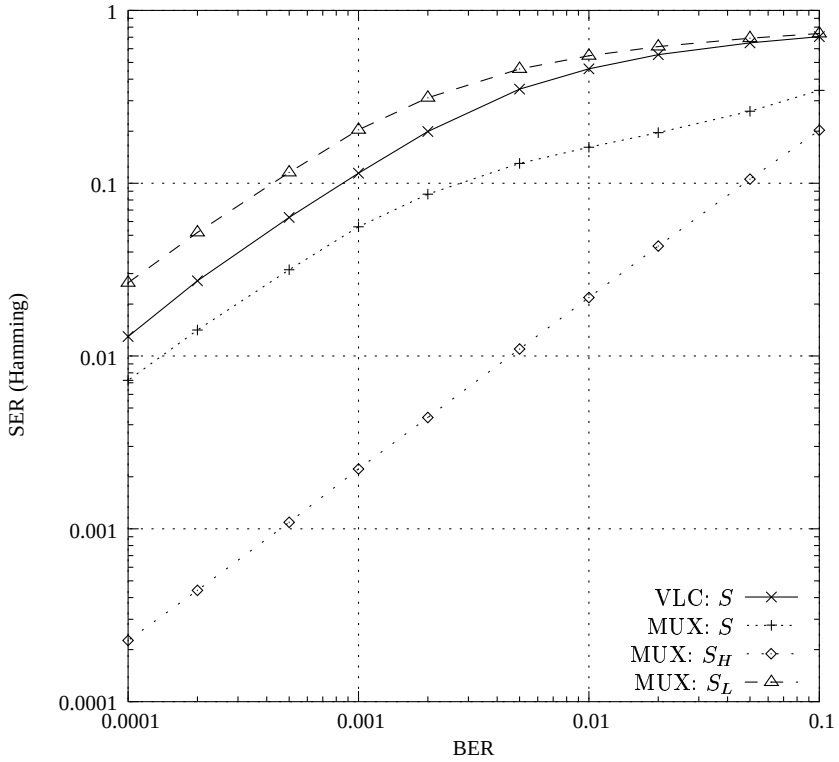


FIG. 3.8 – SER performance of multiplexed codes *vs* Huffman codes.

entire source with both codes (curves VLC : S ; MUX : S). These figures also show the normalized Levenshtein distance and SER obtained on the high (MUX : S_H) and low (MUX : S_L) priority symbols separately. It can be observed that the Levenshtein distance values averaged on the entire sequence are very close with both codes. However, Fig. 3.8 shows the advantage of multiplexed codes if we consider strict-sense synchronization, i.e. the SER or Hamming distance. It can also be observed that significantly lower normalized Levenshtein distance and SER values are obtained for the symbols to be considered as high priority symbols. One can also verify on the curves that the SER approximation given in section 3.9 ($\text{SER}_{\mathbf{S}_H} \approx 2.2 \times \text{BER}$) is valid. The benefit of this inherent unequal error protection feature is better illustrated below with a concrete image coding example.

Experiments with multiplexed codes have been conducted with real sources to show the benefits of both the inherent unequal error protection feature and of the hard synchronization property guaranteed for the high priority symbols. We have considered a simple image coding system where the image is first decomposed into 16 subbands using a two-stage wavelet transform. The low and the high frequency subbands have been quantized respectively on 7 and 4 bits. The high frequencies (i.e.

all the subbands but the low frequency subband) have been encoded with an Huffman algorithm, and then multiplexed into the low frequencies (one subband only) using lexicographic multiplexed codes of parameters $c = 16$ and $f_\nu = 5$. A synchronization marker has been used every four lines for both Huffman codes and the low priority part of the multiplexed codes. The bit rates obtained with Huffman and multiplexed codes are quite similar and given respectively by 1.713 bit per pixel (*bpp*) and 1.712 *bpp*. When considering FLCs, the average bit rate obtained is 6.187 *bpp*. These FLCs are based on the lexicographic index assignment. Fig. 3.9 depicts the PSNR and visual qualities obtained with the three codes. This figure evidences significant improvements both in PSNR and visual quality when using multiplexed codes even in comparison with FLCs and for a similar compression factor as the one obtained with Huffman codes.

3.12 Conclusion

The multiplexed codes described in this chapter avoid the de-synchronization phenomenon for symbols considered to be of high priority, and allow thus the confinement of error propagation to symbols assumed to be of lower priority. The low priority symbols can be pre-encoded with any efficient VLC. A FLC is created for the high priority source, its inherent redundancy being exploited to represent information from the low priority stream. These codes are shown to reach asymptotically the entropy bound for both (low and high priority) sources. Several methods of multiplexing of the two sources are described. Sub-classes of multiplexed codes relying on constrained partitions of the FLC codebook are also introduced. The codes called binary multiplexed codes can be constructed from classical VLC. They thus inherit the compression properties of the VLC considered and allow for multiplexing with very low complexity. Theoretical and simulation results, using both theoretical and real sources, show that these codes are error-resilient at almost no cost in terms of compression efficiency. Another key advantage is that they allow to make use of simple decoding techniques. They hence appear to be excellent alternatives to reversible variable length codes (which suffer from some penalty in terms of compression efficiency, while not avoiding completely error propagation despite a decoding in two passes).

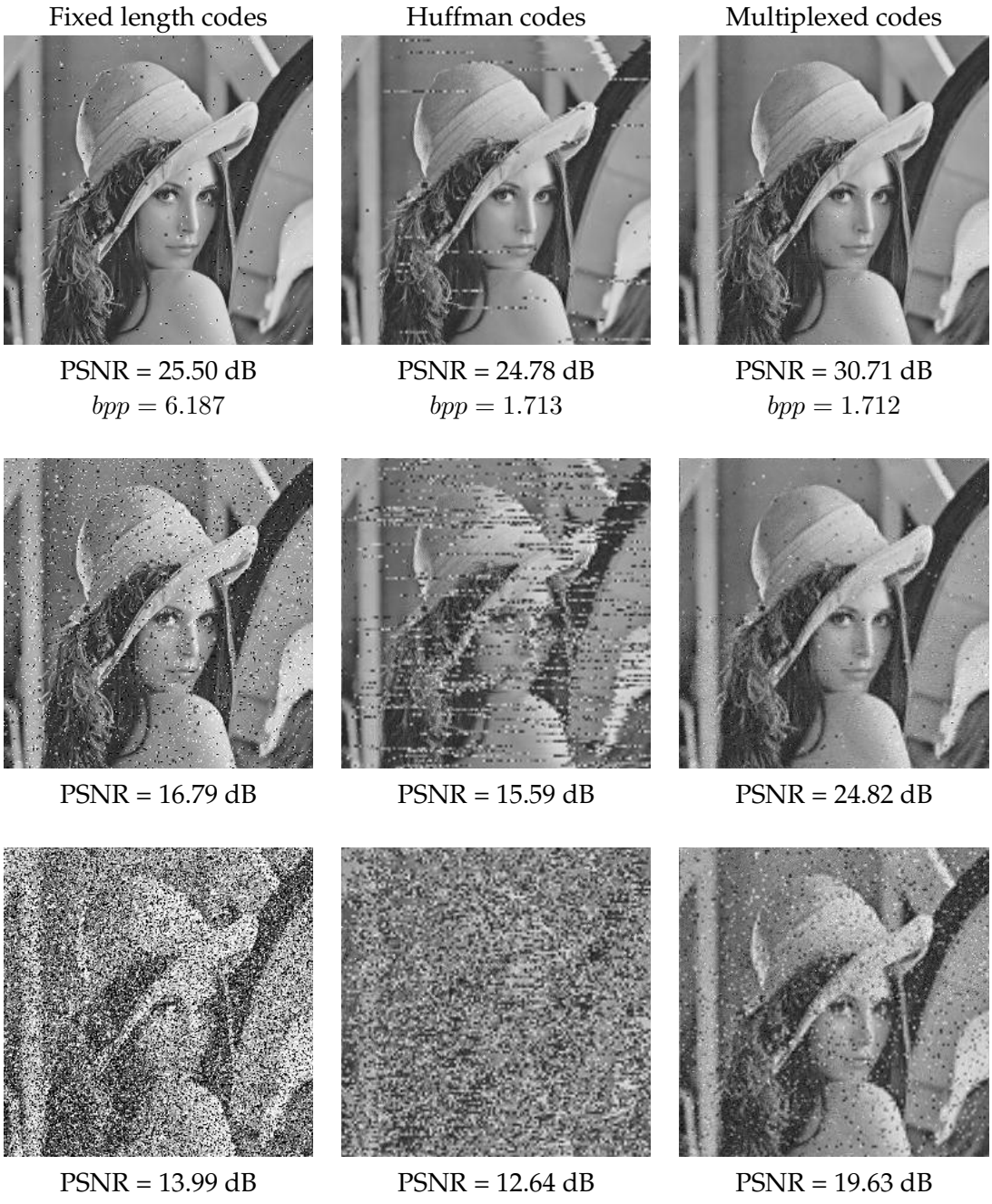


FIG. 3.9 – PSNR performance and visual quality obtained respectively with FLCs, Huffman codes and multiplexed codes. The channel bit error rates are 0.0005 (top images), 0.005 (middle images) and 0.05 (bottom images).

Chapitre 4

First-order multiplexed codes

This chapter has been partially published in [JG03b] and has been presented in [JG04b].

Abstract : This chapter extends the family of error-resilient VLCs described in Chapter 3. The codes introduced can be regarded as generalizations of the multiplexed codes*. They allow the exploitation of first-order source statistics while, at the same time, being resilient to transmission errors. The design principle consists in creating a codebook of fixed length codewords for high priority information, and in using the inherent codebook redundancy to describe low priority information. The FLC codebook is partitioned into equivalence classes according to the conditional probabilities of the high priority source. The error propagation phenomenon, which may result from the memory of the source coder, is controlled by choosing appropriate codebook partitions and IA strategies. In particular, a context-dependent IA strategy, called* crossed-IA*, is described. For the symbol error rate criterion, the approach turns out to maximize the cardinality of a set of codewords, called the code* kernel*, offering synchronization properties. The decoder re-synchronization capability is shown to be further increased by periodic use of memoryless multiplexed codes. Theoretical and practical performance in terms of compression efficiency and error-resilience are analyzed.*

4.1 Introduction

The main drawback of VLCs is their high sensitivity to channel noise : when some bits are altered by the channel, synchronization losses can occur at the receiver. The decoder de-synchronization problem comes from the difficulty in properly segmenting the noisy bitstream into symbols. This segmentation problem can be addressed by introducing a-priori information in the bitstream, taking often the form of synchronization patterns, or of forbidden symbols (e.g. [PSH00]). However, the improved re-synchronization capability is obtained at the expense of loss in compression efficiency. The authors in [RK96] describe a bitstream structure called error-resilient entropy codes (EREC) which allows the reduction of the error propagation phenomenon.

The "multiplexed codes" introduced in Chapter 3 and first presented in [JG03d] make use of the fact that compression systems of real signals generate sources of information with different levels of priority (e.g. texture and motion information for a video signal). Let us recall that these codes are such that the risk of "de-synchronization" is confined to low priority information. The idea consists in creating a fixed length code (FLCs) for the high priority source, and in exploiting the inherent redundancy to represent information of the low priority source. Hence, the high priority source inherits some of the properties of FLCs such as synchronous decoding, random access in the data stream and high error-resilience. These codes naturally offer some form of UEP (to be more precise, unequal error resilience), otherwise often supported via the use of error correcting codes with different rates (e.g. [Hag88]). Here, the property of error-resilience is achieved at a very low cost in terms of redundancy. This redundancy corresponds to the so-called "excess rate" that, depending on the parameters chosen, may result from some compression sub-optimality of the code. This redundancy is thus given by the difference between the actual high priority source description length and its stationary entropy. Note that this "excess rate" becomes asymptotically very small as the length of the FLC codewords increases. However, the codes proposed in Chapter 3 only exploit stationary statistical distributions for the high priority source. In the sequel, we will refer to this family of codes as *memoryless multiplexed codes*.

In this chapter, we address the problem of design of a family of multiplexed codes exploiting first-order statistics for the high priority source, in order to achieve higher compression performance. We focus on first-order statistical models. Similar to *memoryless multiplexed codes*, a set of FLCs is first created for the high priority source ($\mathbf{S}_H$). The inherent redundancy of this code is then exploited to represent information of the low priority source ($\mathbf{S}_L$) which is assumed to be pre-encoded with a classical entropy code (e.g. arithmetic codes). The set of FLCs is then partitioned into equivalence classes according to $\mathbf{S}_H$ conditional probabilities, in order to have an EDL approaching conditional entropy bounds. Theoretical performance *upper bounds* in terms of SER and MSE are then derived for discrete memoryless channels. These bounds provide the SER and MSE performance expectations with hard decision decoding. They can be outperformed when soft decision decoding (e.g., making using of maximum likelihood (ML) or maximum a posteriori estimators) is used instead. As for vector quantizers [ZG90] [Far90], it is shown that the efficiency depends on the codeword IA strategies, i.e., on the construction of the equivalence classes and on the IA used in the different classes. IA strategies aiming at controlling error propagation resulting from erroneous *contexts* as well as the symbols reconstruction error are described. Note that here the term **context** refers to conditioning symbol values. For the SER criterion, a crossed-IA strategy controlling the error propagation turns out to maximize the cardinality of a set of codewords, called the code *kernel*, offering synchronization properties. The multiplexed codes can be decoded at low computational cost by using either hard of soft decision decoding techniques. The complexity of soft decoding of the high prio-

rity source $\mathbf{S}_H$ is linear ($\mathcal{O}(K)$) with respect to the length K of the sequence of symbols. With classical VLCs (e.g. Huffman, arithmetic), the size of the trellis is either in $\mathcal{O}(K)$ or in $\mathcal{O}(K^2)$ [Wei03], depending on whether the termination constraint has to be exploited at the decoder or not. We will however see in Chapter 5 that this complexity of the soft decoding with a length constraint can be substantially decreased. The decoder re-synchronization capability can be further increased by a periodic use of memoryless multiplexed codes to code some symbols of the sequence. An analysis of the trade-off between compression efficiency and re-synchronization capability is given. Theoretical and practical results both in terms of error-resilience and compression efficiency are discussed for different IA strategies and for different decoding techniques. Both hard and soft decision decoding relying on trellis-based decoding techniques (using e.g. the BCJR algorithm [BCJR74]), and considering different estimation criteria (i.e., MAP, MPM, MMSE) have been used. In the sequel, for sake of conciseness, soft decision decoding techniques will be referred to as soft decoding techniques. The SER and SNR performance are evaluated for a binary symmetric channel (BSC) and for an additive white Gaussian noise channel (AWGN). They are compared to state of the art source codes (arithmetic codes). When used in a tandem source-channel coding structure with a $1/2$-rate systematic convolutional code, a coding gain up to 0.75 and 1.5 dB has been achieved with respect to the structure based on arithmetic codes, for sequence lengths of 100 and 1000 symbols respectively.

4.2 Problem statement and Notation

In the following, we reserve capital letters for random variables, and small letters for values of these variables. Bold face characters will be used to denote vectors or sequences. Let $\mathbf{S}_H = (S_1, \ldots, S_t, \ldots, S_{K_H})$ be a sequence of source symbols of high priority taking their values in a finite alphabet $\mathcal{A}$ composed of Ω symbols, $\mathcal{A} = \{a_1, \ldots, a_i, \ldots, a_\Omega\}$. The source $\mathbf{S}_H$ is assumed to be a first-order Markov process. The stationary and conditional probabilities of the source $\mathbf{S}_H$ are respectively denoted $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_i, \ldots, \mu_\Omega)$ and $\boldsymbol{\nu} = (\nu_{1,1}, \ldots, \nu_{i,i'}, \ldots, \nu_{\Omega,\Omega})$, where μ_i stands for the probability $\mathbb{P}(S_t = a_i)$ and $\nu_{i,i'}$ stands for the conditional probability $\mathbb{P}(S_t = a_i | S_{t-1} = a_{i'})$. Let $\mathbf{S}_L$ be a sequence of source symbols of lower priority. This sequence is assumed to be pre-encoded with a classical VLC (e.g. Huffman or arithmetic code). We assume that the bitstream produced by the encoder of the low priority source is a uniformly distributed binary source. This assumption is closely verified if the encoder of the low priority is optimal or close to optimal. Note that the pre-encoding of $\mathbf{S}_L$ can be made conditionally to the realization of $\mathbf{S}_H$. Emitted and received codewords are respectively denoted by the random variables X_t and Y_t. At a given time instant of the sequence of high priority symbols, the variable X_t takes a realization $x_t = x$ where x is a multiplexed codeword as defined in the sequel. The channel is assumed to follow a discrete memoryless channel (DMC) model with transition proba-

bilities $P(Y_t|X_t)$ denoted $R(\cdot,\cdot)$. The reconstructed high priority sequence is denoted $\hat{\mathbf{S}}_H = (\hat{S}_1, \ldots, \hat{S}_t, \ldots, \hat{S}_{K_H})$.

4.3 First-order multiplexed codes

4.3.1 Memoryless multiplexed codes : a review

This section quickly summarizes the prerequisites required for the comprehension of this chapter. A more comprehensive explanation of multiplexed codes is provided in Chapter 3.

Let us denote $\mathcal{X}$ the set of binary codewords of length c. A memoryless multiplexed code is constructed by partitioning the codeword space $\mathcal{X}$ into Ω subsets $\mathcal{C}_i$ of cardinality $|\mathcal{C}_i| = N_i$, called equivalence classes. Note that $\sum_{i=1}^{\Omega} N_i = 2^c$. Each equivalence class $\mathcal{C}_i$ is associated with a symbol a_i of the alphabet $\mathcal{A}$. A codeword $x \in \mathcal{C}_i$ is associated with a symbol a_i of the alphabet $\mathcal{A}$ and to an index value q_i with $0 \leq q_i \leq N_i - 1$. Thus, a multiplexed code is defined by the bijection

$$\begin{array}{ccc} \bigcup_{a_i \in \mathcal{A}} \{\{a_i\} \times \{0, \ldots, N_i - 1\}\} & \rightarrow & \mathcal{X} \\ (a_i, q_i) & \mapsto & x. \end{array} \tag{4.1}$$

Hence, each fixed length codeword x represents jointly a symbol a_i of the high priority source and an index value q_i. Since the value q_i is the realization of an N_i-valued variable, the amount of data described by this index value is $\log_2 N_i$ bits. The expectation of this variable gives the multiplexing capacity per symbol of the high priority sequence. Assuming that this capacity is fully used to describe data of the low priority source, the description length of the symbol a_i is given by $c - \log_2 N_i$ bits. The EDL of the high priority source $\mathbf{S}_H$ is hence given by $\sum_{a_i \in \mathcal{A}} \mu_i \, (c - \log_2 N_i)$. This quantity is equal to the source stationary entropy h^0 if $\forall i,\ c - \log_2 N_i = -\log_2 \mu_i$, i.e. if $N_i = \mu_i \, |\mathcal{X}|$.

Example 4.1: For example, let $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ be the alphabet of the source $\mathbf{S}_H$ with the stationary probabilities given by $\mu_1 = 0.30$, $\mu_2 = 0.43$, $\mu_3 = 0.25$, and $\mu_4 = 0.02$. Table 4.1 gives an example of partitioning of the set $\{0,1\}^3$ into the 4 equivalence classes associated with the alphabet symbols.

The encoding of the two sources proceeds as follows. The realization of the sequence of high priority symbols $\mathbf{S}_H$ is a sequence of symbol values a_i taken from the source alphabet. The low priority bitstream is converted into a sequence of index values q_i. The sequence of pairs (a_i, q_i) provides entries in the multiplexed codes table from which the codewords X_t are extracted. In the previous chapter, two methods have been introduced for the conversion of the low priority source :

class $\mathcal{C}_i$	codeword $c_{i,q}$	a_i	N_i=card($\mathcal{C}_i$)	μ_i	index q_i
$\mathcal{C}_1$	000	a_1	2	0.30	0
	001				1
$\mathcal{C}_2$	010	a_2	3	0.43	0
	011				1
	100				2
$\mathcal{C}_3$	101	a_3	1	0.25	0
	110				1
$\mathcal{C}_4$	111	a_4	1	0.02	0

TAB. 4.1 – An example of memoryless multiplexed codes ($c = 3$).

- The first approach regards the low priority bitstream as an integer which is then decomposed using a hierarchical decomposition. The conversion of this integer into a sequence of states, by itself, has a negligible impact on the compression efficiency (less than 1 bits), as explained in Chapter 3. This approach is optimal in the sense that it allows to have a partition of the FLC codebook which follows closely the PMF of the source $\mathbf{S}_H$.
- A second approach relies on a constrained partition of the FLCs into equivalence classes with cardinalities belonging to a subset of integer values. This approach still leads to a good approximation of the high priority source PMF, hence is quasi-optimal.

In the following either of these two methods applies. We will then only focus on the design of the partition that will depend on first-order statistics of the source $\mathbf{S}_H$.

4.3.2 First-order multiplexed codes : definition

The memoryless multiplexed codes above are constructed for the marginal PMF of the high priority source $\mathbf{S}_H$. However, it is well known that, for Markov sources, higher compression efficiency can be obtained by adapting the codes to higher-order source statistics. We focus here on first-order statistics, however, the approach can be easily extended to higher-orders.

First-order multiplexed codes can thus be constructed for the conditional PMF $\mathbb{P}(S_t = a_i|S_{t-1} = a_{i'})$ of the high priority source. The partition of the set of fixed length codewords $\mathcal{X}$ then becomes conditioned by the realization of the previous symbol $S_{t-1} = a_{i'}$ in the sequence to be encoded. Let $\mathcal{C}_i^{i'}$ and $N_i^{i'}$ respectively denote the equivalence class associated with the symbol a_i and its cardinality when the previous symbol realization is $S_{t-1} = a_{i'}$. The set of codes, for all conditional symbol values from $\mathcal{A}$, defines a so-called *first-order multiplexed code*, referred to as $\mathcal{C}^*$ in the sequel.

			$K_H = 10$	$K_H = 100$	$K_H = 1000$
Multiplexed codes	c=5	theoretical		2.35233	
	c=5	simulated	2.41789	2.35810	2.35295
	c=6	theoretical		2.31168	
	c=6	simulated	2.37636	2.31856	2.31229
Arithmetic codes			2.41026	2.30214	2.29118
entropy h^0				2.46784	
entropy h^1				2.28965	

TAB. 4.2 – Theoretical and simulated mean description length for multiplexed codes and arithmetic codes (Gauss-Markov source with correlation $\rho = 0.5$) for $K_H = 10, 100, 1000$ and for $c = 5, 6$. The theoretical results are obtained from Eqn. 4.3.

The EDL of the code $\mathcal{C}^*$ (for the high priority source $\mathbf{S}_H$) is given by

$$\text{EDL}(\mathcal{C}^*) = \sum_{(a_i, a_{i'}) \in \mathcal{A}^2} \mathbb{P}(S_t = a_i; S_{t-1} = a_{i'})(c - \log_2 N_i^{i'}) \tag{4.2}$$

$$= - \sum_{(a_i, a_{i'}) \in \mathcal{A}^2} \mu_{i'} \, \nu_{i,i'} \log_2 \frac{N_i^{i'}}{|\mathcal{X}|}. \tag{4.3}$$

The EDL of the code $\mathcal{C}^*$ will be equal to the first-order entropy h^1 of the source if and only if

$$\forall (a_i, \, a_{i'}) \in \mathcal{A}^2, \; N_i^{i'} = \nu_{i,i'} \, |\mathcal{X}|. \tag{4.4}$$

Note that the higher compression efficiency may not result in a shorter sequence for $\mathbf{S}_H$, as in classical VLCs, but rather in a higher multiplexing capacity to be used for describing the low priority source $\mathbf{S}_L$. This EDL corresponds to the asymptotic compression performance as the sequence length of the high priority source increases to infinity. In practice, the sequence length has also an impact on the compression performance. First, the first symbol of the sequence has to be encoded with the marginal PMF. Secondly, the conversion of the low priority source into a sequence of states leads to a penalty lying between 0 and 1 bit. Table 4.2 gives EDL values obtained by simulation with different sequence lengths in comparison with the corresponding theoretic EDL values.

The conditioning symbol values are often called contexts in source compression. In practice, when the size of the alphabet $\mathcal{A}$ is large, the number of contexts may be too high leading to a problem of context dilution. This number can be reduced by considering different conditioning values as the same context. The context can then be defined by a function σ of the previous symbol realization, taking its value in the set

of contexts, denoted Σ :

$$\begin{array}{ccc} \mathcal{A} & \rightarrow & \Sigma \\ a_{i'} & \mapsto & \sigma(a_{i'}). \end{array} \tag{4.5}$$

Together with the realization of the symbol $S_t = a_i$ to be encoded, this context identifies the equivalence class to be used, denoted $\mathcal{C}_i^{\sigma(a_{i'})}$. If σ is a constant function, the multiplexed code is a memoryless multiplexed code. In that case, the code is referred to as $\mathcal{C}^{\varnothing}$. The equivalence classes of code $\mathcal{C}^{\varnothing}$ and the corresponding cardinalities are respectively denoted $\mathcal{C}_i^{\varnothing}$ and $N_i^{\varnothing}$, $1 \leq i \leq \Omega$.

4.4 Error-resilience analysis on a DMC

This section analyzes the impact of the use of first-order source statistics on the error-resilience of the code, expressed both in terms of symbol error rate (SER) and of mean square error (MSE) obtained for the high priority source $\mathbf{S}_H$. Let us consider first-order multiplexed codes designed for a Markov source $\mathbf{S}_H$ of transition probabilities $\nu_{i,i'}$. Note first that with first-order multiplexed codes, as with memoryless multiplexed codes, the high priority source does not suffer from de-synchronization problem : the segmentation of the bitstream into high priority source symbols is deterministic. However, erroneous contexts (conditioning values) may lead to error propagation in the decoding of the symbol values.

4.4.1 Modeling the dependencies in the coding and transmission chain

In order to capture the error propagation phenomenon, the SER and MSE performance bounds must be expressed in terms of the PMF $\mathbb{P}(\hat{S}_t|S_t, \hat{S}_{t-1}, S_{t-1})$, or equivalently, with the appropriate re-normalization factor, in terms of $\mathbb{P}(\hat{S}_t, S_t|\hat{S}_{t-1}, S_{t-1})$. The analysis hence requires to consider the global transmission chain formed by the source, the multiplexed source coder, the transmission channel and the decoder. The coding, transmission and decoding chain can be modeled by a process represented by the pair of variables $Z_t = (S_t, \hat{S}_t)$. The process $Z_t = (S_t, \hat{S}_t)$, $t = 1 \dots K$ is a Markov chain. Its transition probability is given by

$$\boldsymbol{\pi} = \begin{pmatrix} \pi(z_0, z_0) & \dots & \pi(z_0, z_{\Omega^2}) \\ \vdots & \ddots & \vdots \\ \pi(z_{\Omega^2}, z_0) & \dots & \pi(z_{\Omega^2}, z_{\Omega^2}) \end{pmatrix}, \tag{4.6}$$

where

$$\pi(z,z') = \mathbb{P}(\hat{S}_t = a_j; S_t = a_i | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) \quad (4.7)$$

$$= \sum_{(X_t,Y_t)\in\mathcal{X}^2} \mathbb{P}(\hat{S}_t = a_j; Y_t; S_t = a_i; X_t | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) \quad (4.8)$$

$$= \sum_{(X_t,Y_t)\in\mathcal{X}^2} \mathbb{P}(S_t = a_i; X_t | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) \times \mathbb{P}(\hat{S}_t = a_j; Y_t | S_t = a_i; X_t; \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}). \quad (4.9)$$

Property 4.1: The transition probability $\pi(z,z')$ is actually given by

$$\pi(z,z') = \frac{\nu_{i,i'}}{N_i^{\sigma(a_{i'})}} \sum_{(X_t,Y_t)\in\mathcal{C}_i^{\sigma(a_{i'})}\times\mathcal{C}_j^{\sigma(a_{j'})}} R(Y_t,X_t), \quad (4.10)$$

where $R(Y_t,X_t)$ represents the channel model, i.e., the probability of receiving the codeword Y_t if the codeword X_t has been emitted.

Proof: Given that $\mathbb{P}(S_t = a_i; X_t | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) = \mathbb{P}(S_t = a_i; X_t | S_{t-1} = a_{i'})$, the first term on the right side of Eqn. 4.9 can be rewritten as

$$\mathbb{P}(S_t = a_i; X_t | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) = \mathbb{P}(X_t | S_t = a_i; S_{t-1} = a_{i'})\, \nu_{i,i'}.$$

In addition, for given symbol realization $S_t = a_i$ and $S_{t-1} = a_{i'}$, the multiplexed codeword X_t is such that $X_t \in \mathcal{C}_i^{\sigma(a_{i'})}$. Therefore, assuming that all the codewords belonging to a given equivalence class have the same probability (which is valid if the low priority source is compressed and converted into a sequence of states in an quasi-optimal manner)

$$\forall X_t \in \mathcal{C}_i^{\sigma(a_{i'})},\ \mathbb{P}(X_t | S_t = a_i; S_{t-1} = a_{i'}) = \mathbb{P}(X_t | X_t \in \mathcal{C}_i^{\sigma(a_{i'})}) = \frac{1}{N_i^{\sigma(a_{i'})}}, \quad (4.11)$$

$$\forall X_t \notin \mathcal{C}_i^{\sigma(a_{i'})},\ \mathbb{P}(X_t | S_t = a_i; S_{t-1} = a_{i'}) = 0. \quad (4.12)$$

From Eqn. 4.11 and Eqn. 4.11 we deduce

$$\forall X_t \in \mathcal{C}_i^{\sigma(a_{i'})},\ \mathbb{P}(S_t = a_i; X_t | \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) = \frac{\nu_{i,i'}}{N_i^{\sigma(a_{i'})}}. \quad (4.13)$$

The other probability of Eqn. 4.9 can be computed as follows, starting with the following simplification :

$$\mathbb{P}(\hat{S}_t = a_j; Y_t | S_t = a_i; X_t; \hat{S}_{t-1} = a_{j'}; S_{t-1} = a_{i'}) = \mathbb{P}(\hat{S}_t = a_j; Y_t | X_t; \hat{S}_{t-1} = a_{j'}). \quad (4.14)$$

Since for a given symbol value $(\hat{S}_t = a_j)$ associated with a multiplexed codeword $Y_t \in \mathcal{C}_j^{\sigma(a_{j'})}$, one has the previous symbol value $\hat{S}_{t-1} = a_{j'}$,

$$\begin{aligned} \forall Y_t \in \mathcal{C}_j^{\sigma(a_{j'})},\ \mathbb{P}(\hat{S}_t = a_j; Y_t | X_t; \hat{S}_{t-1} = a_{j'}) & \\ = \mathbb{P}(Y_t | X_t; S_{t-1} = a_{j'}) & \quad (4.15) \\ = \mathbb{P}(Y_t | X_t) & \\ = R(Y_t, X_t) & \quad (4.16) \end{aligned}$$

$$\forall Y_t \notin \mathcal{C}_{a_j}^{\sigma(a_{j'})},\ \mathbb{P}(\hat{S}_t = a_j; Y_t | X_t; \hat{S}_{t-1} = a_{j'}) = 0. \quad (4.17)$$

Then from Eqn. 4.12, Eqn. 4.13, Eqn. 4.16 and Eqn. 4.17, one can finally deduce

$$\pi(z, z') = \frac{\nu_{i,i'}}{N_i^{\sigma(a_{i'})}} \sum_{(X_t, Y_t) \in \mathcal{C}_i^{\sigma(a_{i'})} \times \mathcal{C}_j^{\sigma(a_{j'})}} R(Y_t, X_t). \quad (4.18)$$

■

The entity $\nu_{i,i'}$ represents the conditional probability of the high priority source, and $\mathcal{C}_i^{\sigma(a_{i'})}$ the corresponding equivalence class of cardinality $N_i^{\sigma(a_{i'})}$.

4.4.2 Asymptotic SER and MSE bounds

For a sequence of infinite length, the Markov process Z_t converges toward a stationary state of probability $\mathbb{P}(Z_t) = \mathbb{P}(S_t; \hat{S}_t)$. The corresponding asymptotic SER and MSE performance bounds are thus functions of the transition PMF $\mathbb{P}(\hat{S}_t | S_t)$, or equivalently of the joint probability $\mathbb{P}(S_t; \hat{S}_t)$. The joint PMF $\mathbb{P}(S_t; \hat{S}_t)$ can be regarded as the stationary probability of the Markov process defined by the pair of variables $Z_t = (S_t, \hat{S}_t)$. Assuming that Z_t is irreductible and aperiodic, the stationary probability is given by the solution of the equation $\lambda = \boldsymbol{\pi}\lambda$ (see, e.g., Chapter 4 of [CT91]). In other terms, this stationary PMF is obtained by processing the normalized eigenvector associated with the eigenvalue 1 of the transition matrix $\boldsymbol{\pi}$. In the following, this eigenvector is denoted

$$\boldsymbol{\lambda}_\infty = (\lambda_\infty(a_1, a_1), \ldots, \lambda_\infty(a_1, a_\Omega), \ldots, \lambda_\infty(a_i, a_j), \ldots, \lambda_\infty(a_\Omega, a_\Omega))^T.$$

This PMF $\boldsymbol{\lambda}_\infty$ verifies the relation $\boldsymbol{\lambda}_\infty = \boldsymbol{\pi}\,\boldsymbol{\lambda}_\infty$. The asymptotic value of the SER can be expressed in terms of the summation of probabilities over product states $z = (a_i, a_i)$, i.e.

$$\begin{aligned} \mathrm{SER}_{\mathcal{C}^*} &= 1 - \sum_{a_i \in \mathcal{A}} \lambda_\infty(a_i, a_i) & (4.19) \\ &= 1 - \boldsymbol{\delta}^T \boldsymbol{\lambda}_\infty, & (4.20) \end{aligned}$$

where the vector $\boldsymbol{\delta}$ denotes a vector of $\mathcal{A} \times \mathcal{A}$ such that $\delta(a_i, a_j) = 1$ if $a_i = a_j$, and $\delta(a_i, a_j) = 0$ otherwise. Similarly, the asymptotic performance of the MSE is given by

$$\text{MSE}_{\mathcal{C}^*} = \boldsymbol{\Delta}^T \boldsymbol{\lambda}_\infty, \tag{4.21}$$

where the distortion vector $\boldsymbol{\Delta}$ is defined such that $\Delta(a_i, a_j) = ||a_i - a_j||^2$. These bounds have been verified experimentally : as the number of channel realizations increases, the experimental SER and MSE values asymptotically converge toward the theoretical bounds given in Eqn.4.20 and Eqn.4.21.

4.4.3 Impact of the IA on error-resilience

The assignment of multiplexed codewords to the different equivalence classes for all the possible contexts (i.e. finding the partition) is referred to in the sequel as IA. In the context of vector quantization, it has been shown that error-resilience can be improved by modifying the binary labeling of codewords. To improve an existing IA, the binary switching algorithm [ZG90] allows one to find a local optimum. Simulated annealing [GHSW87, Far90] aims at finding a global optimum, by avoiding local optima in a statistical manner. It generally provides better results than the binary switching algorithm. It has also been shown that channel-optimized vector quantization [Far90] can improve the overall rate-distortion efficiency, By *overall* we mean that the distortion includes both the source quantization noise and the channel noise. but in that case the designed quantizer may not be optimum for noiseless channels. In this chapter, the codebooks are assumed to be optimum, in terms of compression, i.e. they approximatively satisfy Eqn. 4.4. Consequently, the analysis is restricted to the IA of binary codewords. The IA is observed to have a major impact on the multiplexed codes error-resilience, as shown in Fig. 4.1. The optimal IA depends on channel and source properties. However, it may also depend on the performance measure used (e.g. SER or SNR). Fig. 4.1 shows that an IA based on Gray codes has better SER performance than IA with lexicographic codes. However, the lexicographic IA outperforms the Gray code when the SNR performance measure is used.

4.5 Code design : crossed-IA and code kernel

The design of a first-order multiplexed code can be decomposed into the three following steps :

1. the codeword length c is first chosen so that the ratio between the EDL of the source S_H and c (that is, $\frac{EDL}{c}$) corresponds to the expected ratio between the number of high priority and of low priority bits required to encode the high priority and the low priority sequences of symbols, respectively.

2. For each possible context, the cardinalities of equivalence classes are chosen according to the probability of symbols knowing the given context, as given in Eqn. 4.4. This step has a strong impact on the compression efficiency of the code.
3. The last step is the IA. This step impacts the error-resilience, but not the compression efficiency of the code.

The two first steps have been adressed in detail in Chapter 3 and are briefly revised in Section 4.3.1. The methods and analysis proposed for Step 1 and Step 2 still apply to first-order multiplexed codes. In this section, we focus on the last step, which turns out to be a key issue in a context of error-resilience. For memoryless multiplexed codes, finding the optimal IA has a prohibitive computational cost : the number of possible partitions of $\mathcal{X}$ into Ω subsets of cardinality N_i ($1 \leq i \leq \Omega$) is given by $\frac{|\mathcal{X}|!}{\prod_{i=1}^{\Omega} N_i!}$. This number corresponds to the number of all possible distinct partitions. For first-order multiplexed codes, the number of possible partitions is even higher and given by

$$\prod_{\eta \in \Sigma} \left(\frac{|\mathcal{X}|!}{\prod_{i=1}^{\Omega} N_i^{\eta}!} \right). \tag{4.22}$$

The IA conditioned by a given context must take into account the IA conditioned by the other contexts. That is why the IA is referred to as crossed-IA in the sequel.

The IA can in addition be designed in order to optimize different criteria, e.g. SER or MSE with performance bounds respectively given in Eqn. 4.20 and Eqn. 4.21, assuming hard-decision decoding. The optimal IA is hence given by the set of partitions (one per context) and corresponding codeword indexing leading to the minimum argument of these expressions. Each step of the IA optimization procedure according to Eqn. 4.20 and Eqn. 4.21 requires the computation of eigenvalues of matrices of size $\Omega^2 \times \Omega^2$. The resulting complexity is not easily tractable. One can design crossed-IA strategies allowing to obtain good performance with a much reduced complexity. In the sequel, a specific approach of crossed-IA will be designed. For sake of clarity, this approach will be also called crossed-IA.

4.5.1 Crossed-IA strategy

Simplified optimization criteria controlling 1) error propagation due to erroneous contexts and 2) the reconstruction error of the current symbol can be considered. We first focus on reducing the error propagation effect by constructing the partition in such a way that the decoding of a symbol will result in a correct context even if the current context is erroneous.

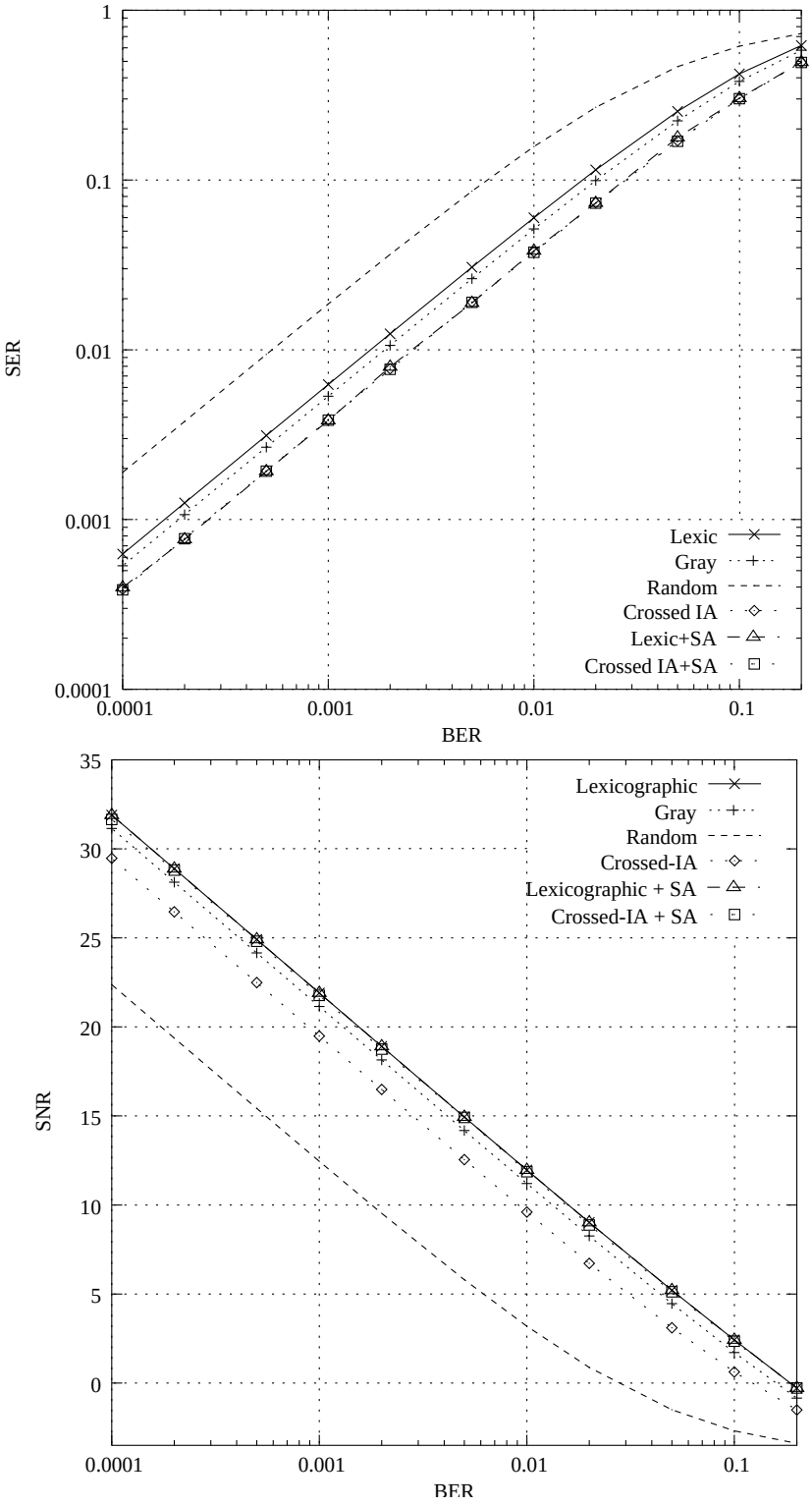


FIG. 4.1 – Theoretical SER and SNR performance obtained with different IA strategies. The source considered is a Gauss-Markov source ($\rho = 0.5$) quantized on 8 levels. The first-order multiplexed code used is characterized by $c = 5$ and 8 contexts.

4.5.1.1 Reduction of error propagation

The problem of IA at a given time instant is then of trying to maximize the probability that the next symbol will be decoded with a correct context. Let us denote $\mathbb{P}(\eta)$ the probability of having the context η at a symbol clock instant t. This probability is deduced from the source stationary probabilities by

$$\mathbb{P}(\eta) = \sum_{a_i \in \sigma^{-1}(\eta)} \mathbb{P}(S_t = a_i), \tag{4.23}$$

where $\sigma^{-1}(\eta) = \{a_i : \sigma(a_i) = \eta\}$. Now, let us denote a_x^η the symbol of $\mathcal{A}$ represented by the codeword x if the current state of the decoder is the context η. If the received codeword is correct but the context is erroneous, for example $\tilde{\eta}$ instead of η, the context used for the next symbol will be erroneous if and only if $\sigma(a_x^\eta) \neq \sigma(a_x^{\tilde{\eta}})$. Then, we can derive the following similarity measure, which applies to codewords :

$$d_{\mathcal{X}}(x) = \sum_{(\eta,\tilde{\eta}) \in \Sigma^2} \mathbb{P}(\eta)\, \mathbb{P}(\tilde{\eta}|\eta)\, \delta(\sigma(a_x^\eta), \sigma(a_x^{\tilde{\eta}})), \tag{4.24}$$

where $\delta(.,.)$ denotes the Kronecker operator. The measure $d_{\mathcal{X}}(x)$ represents the probability that decoding the codeword x leads to a correct context for subsequent symbols, even if the current context is in error. In order to reduce the computational cost, i.e., to avoid seeking eigenvectors of large matrices, we assume that $\mathbb{P}(\tilde{\eta}|\eta) \approx \mathbb{P}(\tilde{\eta})$. Although this approximation is biased, it has been experimentally observed that it does not have a noticeable impact on the error-resilience property of the code. This measure can be extended to the whole set of codewords, which leads to

$$d_{\mathcal{X}}(\mathcal{C}^*) = \sum_{x \in \mathcal{X}} \mathbb{P}(x)\, d_{\mathcal{X}}(x) \approx \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} d_{\mathcal{X}}(x). \tag{4.25}$$

The IA can then be chosen so that the measure $d_{\mathcal{X}}(\mathcal{C}^*)$ is optimized. The IA algorithm then proceeds as follows : the IA is first initialized by taking e.g. a lexicographic order. The algorithm then proceeds with permutations of the different symbols across the equivalence classes, and this for the different contexts. If a permutation leads to an increase of $d_{\mathcal{X}}(\mathcal{C}^*)$, then it is retained. The algorithm stops when no further increase of $d_{\mathcal{X}}(\mathcal{C}^*)$ is observed. The underlying optimization principle is close to the binary switching algorithm [ZG90], in so far as a local optimum of Eqn. 4.25 is searched using symbol permutations. A simulated annealing (SA) can also be applied. However, experimentally, we did not observe any significant improvement in terms of convergence of the final value $d_{\mathcal{X}}(\mathcal{C}^*)$.

4.5.1.2 Constrained optimization of the IA

The value $d_{\mathcal{X}}(\mathcal{C}^*)$ only depends on the assignment of symbols a_i of $\mathcal{A}$ to the different classes, and this taking into account the different possible contexts. It does not

take into account the channel characteristics, nor the binary representations of the codewords. The permutation, for all the contexts, of the binary representations of two codewords[1] x_1 and x_2 does not modify the value $d_{\mathcal{X}}(\mathcal{C}^*)$. One can then choose the binary representations of the different codewords in order to reduce the SER (Eqn. 4.20) or the MSE (Eqn. 4.21). The above optimization of the crossed-IA can be carried out using an SA algorithm.

The SA algorithm can also be processed such that the perturbations correspond to an arbitrary choice of the contexts and the codewords to be switched. The performance of this SA algorithm strongly depends on the initial choices of the parameters. We first applied the SA algorithm using a lexicographic code as the initial state. However, with the SER criterion, the resulting codes were, most of the time, less error-resilient than the one obtained with the sequential approach described in sections 4.5.1.1 and 4.5.1.2. Hence, in the sequel, we have chosen the code resulting from the approach referred to as crossed-IA, and described in sections 4.5.1.1 and 4.5.1.2, as the initialization of the SA algorithm.

4.5.2 Definition and properties of the kernel

The crossed-IA strategy according to the SER criterion actually turns out to construct a subset of codewords, referred to as *the kernel* of the code with the following properties.

Definition 1 : the kernel $\mathcal{K}$ of a multiplexed code $\mathcal{C}^*$ is the set of codewords of $\mathcal{X}$ such that

$$x \in \mathcal{K} \text{ if and only if } \exists\, a_i \in \mathcal{A} \,/\, \forall\, \eta \in \Sigma,\ x \in \mathcal{C}_i^{\eta}. \tag{4.26}$$

In other words, a codeword that belongs to the kernel always represents the same symbol of $\mathcal{A}$, whatever the context. Thus, the kernel can be partitioned into Ω sets of codewords, as

$$\mathcal{K} = \bigcup_{a_i \in \mathcal{A}} \mathcal{K}_i, \tag{4.27}$$

where $\mathcal{K}_i$ is the set of codewords of the kernel that represents the symbol a_i for every context. For a memoryless multiplexed code, we have obviously $|\mathcal{K}_i| = N_i$ and $\mathcal{K} = \mathcal{X}$. For a first-order multiplexed code, the way a codeword of the kernel is decoded does not depend on past realizations. Thus, these codewords act as *hard synchronization* points. The probability to encode a symbol with a codeword of the kernel is given by

$$\mathbb{P}(X_t \in \mathcal{K}) = \sum_{(a_i, a_{i'}) \in \mathcal{A}^2} \nu_{i,i'} \frac{|\mathcal{K}_i|}{N_i^{\sigma(a_{i'})}} \approx \frac{|\mathcal{K}|}{|\mathcal{X}|}. \tag{4.28}$$

[1] which is equivalent to performing, for each context η, the permutation of the symbols $a_{x_1}^{\eta}$ and $a_{x_2}^{\eta}$.

The proposed approximation is deduced from the approximation of $N_i^{i'}$ given in Eqn. 4.4. It is valid if the PMF $\mathbb{P}(x)$ over codewords is almost uniform (i.e. $\forall x,\ \mathbb{P}(x) \approx |\mathcal{X}|^{-1}$).

4.5.3 Maximum cardinality of the kernel

The cardinalities $N_i^{\sigma(a_{i'})}$ are first chosen in order to minimize the EDL given by Eqn. 4.3. Assuming that these values have already been found, one can then try to maximize the rate of *hard synchronization* points, i.e., to maximize the cardinality $\mathcal{K}$ of the kernel. From Definition 1, we deduce that

$$\mathcal{K}_i = \mathcal{K} \bigcap_{a_{i'} \in \mathcal{A}} \mathcal{C}_i^{\sigma(a_{i'})}, \tag{4.29}$$

which leads to

$$|\mathcal{K}_i| \leq \min_{a_{i'} \in \mathcal{A}} N_i^{\sigma(a_{i'})}. \tag{4.30}$$

This means that, for each symbol value a_i of the high priority source $\mathbf{S}_H$, a maximum number of $\min_{a_{i'} \in \mathcal{A}} N_i^{\sigma(a_{i'})}$ codewords are assigned to classes $\mathcal{C}_i^{\sigma(a_{i'})}$, independently of the context values (or past symbol realizations). The equality can be obtained with some appropriate IA. In particular, we observed that the crossed-IA strategy with the SER criterion leads to maximize the kernel cardinality. The code kernel property can be used for random data access.

4.6 Soft decoding of multiplexed codes

Soft decoding techniques based on Bayesian estimation can be used to further improve the SER and SNR performance of multiplexed codes. The BCJR algorithm [BCJR74] directly applies provided that the forward and backward recursions are initialized in an appropriate manner as described below. This algorithm then allows to estimate the probabilities $\mathbb{P}(\mathbf{S}_H|\mathbf{S}_1, \ldots, \mathbf{S}_{K_H})$ and $\mathbb{P}(S_t|\mathbf{S}_t)$ needed for deriving the symbol or sequence MAP or MMSE estimates.

4.6.1 Adaptation of the BCJR algorithm for multiplexed codes

Let us consider first the decoding of the high priority source which is assumed to be a Markov source. One could consider a joint decoding of both sources $(\mathbf{S}_H, \mathbf{S}_L)$, however at the expense of a prohibitive computational cost. Since $\mathbf{S}_H$ defines a Markov process, a state of the trellis is fully identified by the source realization, as for the case of FLCs. Thus, the section of the trellis is of cardinality Ω. Therefore, unlike VLCs, an optimal[2] trellis decoding based on the BCJR algorithm can be performed with a

[2]optimal in the sense that the so-called "excess-rate" is fully exploited : no pruning is done there.

complexity in $O(K_H)$, which allows for soft decoding of very large sequences. A detailed analysis of this complexity is provided in section 4.6.1.2. The BCJR algorithm proceeds with the estimation of the probabilities $\mathbb{P}(S_t = a_i|Y_1; \ldots; Y_K)$, knowing

- the Markov source transitions probabilities, i.e., $\nu_{i,i'} = \mathbb{P}(S_t = a_i|S_{t-1} = a_{i'})$,
- the probabilities $\mathbb{P}(X_t = x|S_{t-1} = a_{i'}; S_t = a_i)$, where X_t denotes the codeword emitted at the instant t corresponding to the symbol S_t. Since the PMF of the codewords of a given equivalence class is uniform, we have

$$\mathbb{P}(X_t = x|S_{t-1} = a_{i'}; S_t = a_i) = \begin{cases} \frac{1}{N_i^{\sigma(a_{i'})}} & \text{if } x \in \mathcal{C}_i^{\sigma(a_{i'})}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

- The channel transition probabilities $\mathbb{P}(Y_t = y|X_t = x)$, which are assumed to follow a DMC model.

In the following, the probabilities involved in the two recursions are denoted using the same notations as in [BCJR74], namely

$$\begin{array}{rcl} \alpha_t(i) & = & \mathbb{P}(S_t = a_i; Y_1 = y_1; \ldots; Y^t = y_t), \\ \beta_t(i) & = & \mathbb{P}(Y_{t+1} = y_{t+1}; \ldots; Y_K = y_K|S_t = a_i), \\ \gamma_t(i', i) & = & \mathbb{P}(S_t = a_i; Y_t = y_t|S_{t-1} = a_{i'}). \end{array} \tag{4.32}$$

The BCJR algorithm can also compute the probabilities $\mathbb{P}(\mathbf{s}_H|\mathbf{s}_1, \ldots, \mathbf{s}_{K_H})$. This probability is used by the MAP algorithm, but is not required for MPM (symbol MAP) and MMSE algorithms.

4.6.1.1 Initialization of forward and backward recursions

Here, the initial and terminating states cannot be set to 0, as in [BCJR74]. In contrast, the probability $\alpha_1(m) = \mathbb{P}(S_t; Y_1)$, is computed from the DMC channel model and the memoryless code used for the first symbol as

$$\begin{aligned} \alpha_1(i) &= \mathbb{P}(Y_1 = y_1|S_1 = a_i)\, \mathbb{P}(S_1 = a_i) \\ &= \mu_i \sum_{x \in \mathcal{C}_{a_i}^{\varnothing}} R(y_1|x). \end{aligned} \tag{4.33}$$

Similarly, since for codes with an EDL close to the first-order entropy, the PMF of codewords is close to uniform, the probability $\beta_K(i)$, of each state i in the last section of the trellis, can be chosen as $\beta_K(i) = |\mathcal{X}|^{-1}$.

4.6.1.2 Complexity analysis

The computational cost of the BCJR algorithm applied to the decoding of the multiplexed codes is in $O(K_H)$ and can be expressed as $(C_m + C_a)\, K_H$, where C_m and C_a respectively denote the number of floating point multiplications and additions involved per symbol clock instant t. The factor C_m can be decomposed as $C_m = C_m(\gamma) + C_m(\alpha) + C_m(\beta) + C_m(\mathbb{P}(S_t|Y_1;\ldots;Y_{K_H}))$, where the variable $C_m(\square)$ denotes the computing cost per symbol clock instant for each variable $\square$. The same decomposition applies to C_a. The overall computational complexity is given by

$$C_m = \Omega\,(\,|\mathcal{X}| + 3\,\Omega + 1\,), \tag{4.34}$$

$$C_a = \Omega\,(\,|\mathcal{X}| + 2\,\Omega\,). \tag{4.35}$$

Proof: To begin with, the probability $\gamma_t(i', i)$ is computed as

$$\gamma_t(i', i) = \nu_{i,i'} \sum_{x \in \mathcal{X}} \mathbb{P}(X_t = x | S_{t-1} = a_{i'}, S_t = a_i)\, R(Y_t|x). \tag{4.36}$$

According to Eqn. 4.31, the probability $\mathbb{P}(X_t = x|S_{t-1} = a_{i'}; S_t = a_i)$ is not null if and only if $x \in \mathcal{C}_i^{\sigma i'}$. This allows to restrict the summation to the set of codewords $\mathcal{C}_i^{\sigma i'}$ in Eqn. 4.36. The cardinality of this equivalence class being $N_i^{\sigma(i')}$, the numbers of multiplications and additions involved in the calculation of $\gamma_t(i', i)$ for each tuple $(a_i, a_i') \in \mathcal{A}^2$ are respectively given[3] by $N_i^{\sigma(i')} + 1$ and $N_i^{\sigma(i')}$. Since $\forall a_{i'},\ \sum_{a_i \in A} N_i^{\sigma(a_{i'})} = |\mathcal{X}|$, the numbers of multiplications and additions for the variable γ are given by

$$C_m(\gamma) = (|\mathcal{X}| + \Omega)\,\Omega, \quad C_a(\gamma) = |\mathcal{X}|\,\Omega, \tag{4.37}$$

and by

$$\begin{array}{lll} C_m(\alpha) = \Omega^2, & C_m(\beta) = \Omega^2, & C_m(\mathbb{P}(S_t|Y_1,\ldots,Y_K)) = \Omega, \\ C_a(\alpha) = \Omega^2, & C_a(\beta) = \Omega^2, & C_a(\mathbb{P}(S_t|Y_1,\ldots,Y_K)) = 0, \end{array} \tag{4.38}$$

for α, β and λ respectively. The proposed overall computational complexity are subsenquently obtained. ■

Note that, since floating point numbers in machines are represented with finite precision, some re-normalization of probabilities α and β may be required, at the expense of a slight increase of the computational complexity. For example, the re-normalization processed such that $\forall t,\ \sum_{a_i \in \mathcal{A}} \alpha_t(i) = 1$ and $\forall t,\ \sum_{a_i \in \mathcal{A}} \beta_t(i) = 1$ leads to an increase of C_m and C_a by $2\,\Omega$. Note also that the computational cost above is slightly increased if the MAP algorithm is used, due to the additional estimation of $\mathbb{P}(S_{t-1} = m'; S_t = m; Y_1; \ldots; Y_{K_H})$ and the subsequent processing of the Viterbi algorithm [Vit67].

[3] An addition can be avoided with an appropriate implementation.

4.6.2 Estimation criteria : MAP, MPM and MMSE

For the cost function $C(k,j) = 1 - \delta_{k,j}$, the MAP criterion (maximum *a posteriori*) corresponds to the optimal Bayesian estimation[4] of a process S based on all available measurements Y : the MAP decoding problem consists then in computing

$$\hat{S} = \hat{S}_1 \dots \hat{S}_K = \arg \max_{s_1 \dots s_{K_H}} \mathbb{P}(S_1 = s1; \dots; S_{K_H} = s_{K_H} | Y_1, \dots, Y_{K_H}), \tag{4.39}$$

where $s_1, \dots, s_K$ are the realizations of the symbols $S_1, \dots, S_{K_H}$ and take their values in the alphabet $\mathcal{A}$. The optimization is over all possible *sequences* s. For the SER criterion, the optimal sequence $\tilde{S}$ can be obtained by gathering local estimates $\tilde{S}_t$ defined by

$$\tilde{S}_t = \arg \max_{a_i \in \mathcal{A}} \mathbb{P}(S_t = a_i | Y_1, \dots, Y_{K_H}), \tag{4.40}$$

which are the MPM (maximum of posterior marginals) estimates of each hidden state of the Markov chain. However, if the mean square error (MSE) is the performance measure, estimation based on probabilities only are sub-optimal. In this case, the corresponding (sequence and symbol-based [MP98]) optimal decoders are decoders that compute conditional means or minimum MSE (MMSE). In the case of the symbol-based MMSE decoder, the conditional mean is defined as

$$\overline{S}_t = \sum_{a_i \in \mathcal{A}} \mathbb{P}(S_t = a_i | Y_1, \dots, Y_{K_H})\, a_i, \tag{4.41}$$

with $\mathbb{P}(S_t | Y_1, \dots, Y_{K_H})$ sub-product of the BCJR algorithm. The decoder seeks a sequence of reproductions that will minimize the expected distortion, given the sequence of observations.

4.7 Hard and soft re-synchronization of contexts

The SER and SNR performance can be further improved at the expense of a slight decrease in compression efficiency, by encoding some symbols at known positions with a memoryless multiplexed code rather than with a first-order code. The codewords of the memoryless multiplexed code then act as *hard synchronization* points. One can also reduce the number of contexts in order to increase the source "excess-rate" which can in turn be exploited by using the MPM or MMSE decoding algorithms. Note that the BCJR algorithm [BCJR74] directly applies. Note also that, unlike VLCs, an optimal[5] trellis decoding based on the BCJR algorithm can be performed with a complexity in $\mathcal{O}(K_H)$, which allows for soft decoding of very large sequences.

[4] In that case, the average Bayes risk is the probability of error.

[5] optimal in the sense that the "excess-rate" is fully exploited ; no pruning is done.

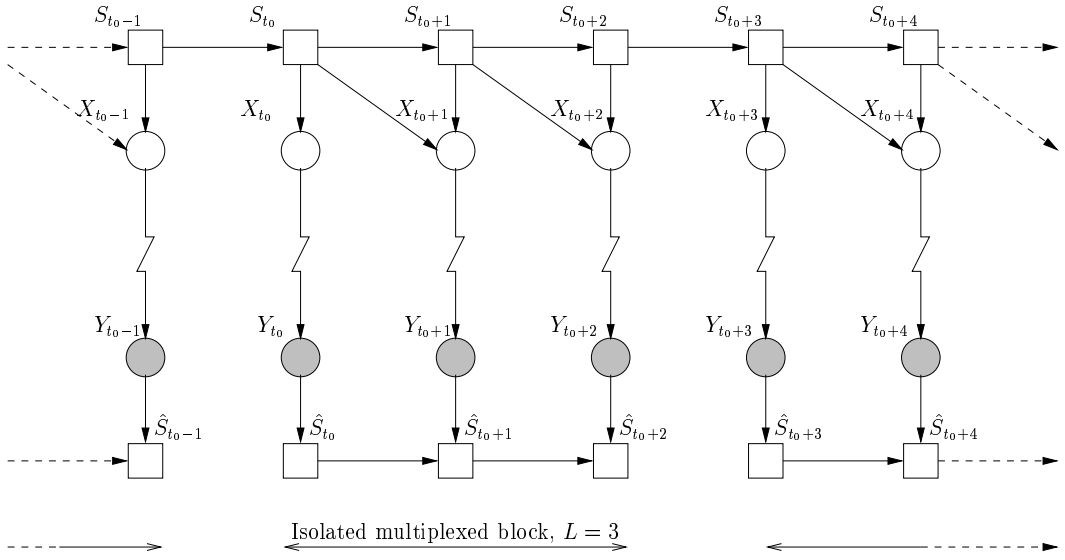


FIG. 4.2 – Hard synchronization of the decoder using memoryless multiplexed codewords ($\mathcal{C}^{\varnothing}$) every 3 symbols ($L = 3$). The decoded symbols $\hat{S}_{t_0}$ and $\hat{S}_{t_0+3}$ depend only on the received codewords Y_{t_0} and Y_{t_0+3}.

4.7.1 Error-resilience analysis for finite length sequences

Let $\lambda_t(i,j)$ denote the probability $\mathbb{P}_t(S_t = a_i; \hat{S}_t = a_j)$ that the event $(S_t = a_i; \hat{S}_t = a_j)$ occurs at a given symbol clock instant t. Let $\boldsymbol{\lambda}_t$ be the vector of probabilities that the coder-decoder product model is in the different states of the state space $\mathcal{A}^2$. Since we assume that the first symbol is encoded with a multiplexed code $\mathcal{C}^{\varnothing}$ (according to the stationary probability of the source $\mathbf{S}_H$), the related vector $\boldsymbol{\lambda}_1$ is denoted $\boldsymbol{\lambda}_{\varnothing}$. Its components $\lambda_{\varnothing}(i,j)$ are deduced from the stationary PMF $\boldsymbol{\mu}$ of the source $\mathbf{S}_H$ and from the channel model as

$$\begin{aligned}
\forall (i,j) \in \mathcal{A}^2,\ \lambda_{\varnothing}(i,j) &= \mathbb{P}(S_t = a_i; \hat{S}_t = a_j) && (4.42)\\
&= \sum_{(X_t,Y_t)\in \mathcal{C}_i^{\varnothing}\times\mathcal{C}_j^{\varnothing}} \mathbb{P}(Y_t; X_t) && (4.43)\\
&= \sum_{(X_t,Y_t)\in \mathcal{C}_i^{\varnothing}\times\mathcal{C}_j^{\varnothing}} \mathbb{P}(Y_t|X_t)\,\mathbb{P}(X_t) && (4.44)\\
&= \frac{\mu_i}{N_i^{\varnothing}} \sum_{(X_t,Y_t)\in \mathcal{C}_i^{\varnothing}\times\mathcal{C}_j^{\varnothing}} R(Y_t, X_t). && (4.45)
\end{aligned}$$

For the symbol clock instant $t = 2$, the vector of probabilities $\boldsymbol{\lambda}_2$ is given by $\boldsymbol{\pi}\boldsymbol{\lambda}_{\varnothing}$. Similarly, the general expression of the state for any given symbol clock instant t is

given by

$$\boldsymbol{\lambda}_t = \boldsymbol{\pi}^{t-1}\boldsymbol{\lambda}_{\varnothing}. \tag{4.46}$$

Then, the SER of a sequence of K codewords is given by the average of symbol error rates over the sequence, i.e.

$$\mathrm{SER}_{\mathcal{C}^*,K} = 1 - \frac{1}{K}\sum_{t=1}^{K} \boldsymbol{\delta}^T \boldsymbol{\lambda}_t \tag{4.47}$$

$$= 1 - \frac{1}{K}\boldsymbol{\delta}^T \left(\sum_{t=1}^{K} \boldsymbol{\pi}^{t-1}\right).\boldsymbol{\lambda}_{\varnothing} \tag{4.48}$$

Similarly, the average mean square error for the entire sequence is given by

$$\mathrm{MSE}_{\mathcal{C}^*,K} = \frac{1}{K}\boldsymbol{\Delta}^T \left(\sum_{t=1}^{K} \boldsymbol{\pi}^{t-1}\right) \boldsymbol{\lambda}_{\varnothing}. \tag{4.49}$$

4.7.2 Synchronization using memoryless multiplexed codes

In order to further reduce error propagation, one can encode symbols at a priori known positions[6], with a memoryless multiplexed code constructed from the stationary probability $\boldsymbol{\mu}$. This code has already been introduced in section 4.7.1, and is referred to as $\mathcal{C}^{\varnothing}$. It asymptotically reaches, for high values of c, the stationary entropy of the source. Here, we restrict the analysis to the case where this code is used at symbol clock positions t such that $t \in L\,\mathbb{N}$, where $L \in \mathbb{N}^+ - \{0\}$. The corresponding code is referred to as $\mathcal{C}^{*,L}$. Every L symbols, the code $\mathcal{C}^{\varnothing}$ acts as a *hard re-synchronization* point as depicted in figure 4.2. The rate of symbols encoded using the conditional probabilities is given by $\frac{L-1}{L}$, which leads to the following compression efficiency for $\mathcal{C}^{*,L}$:

$$\mathrm{EDL}(\mathcal{C}^{*,L}) = \frac{1}{L}\left(\mathrm{EDL}(\mathcal{C}^{\varnothing}) + (L-1)\,\mathrm{EDL}(\mathcal{C}^{*})\right). \tag{4.50}$$

Since some storage capacity can be lost during the conversion of the pre-encoded low priority source $\mathbf{S}_L$ into a finite sequence of states, this conversion is performed for the whole sequence, according to the values $N_i^{\varnothing}$ and $N_i^{\sigma(i')}$ deduced from the code $\mathcal{C}^{*,L}$ and the realization of $\mathbf{S}_H$. Now, let us recall that the zero order and first-order entropies of the source $\mathbf{S}_H$ are respectively denoted h^0 and h^1. If we assume that the codeword length c allows for a good precision in the stationary and conditional PMF approximations, the compression efficiency approximately satisfies

$$\mathrm{EDL}(\mathcal{C}^{*,L}) = \frac{h^0 + (L-1)h^1}{L}. \tag{4.51}$$

[6] equivalently, fixed bit positions

Each codeword block of length L can be seen as a sequence of multiplexed codewords of finite length L : the vector $\lambda_{\varnothing}$ provides the probabilities to observe the event $(S_t = a_i) \wedge (\hat{S}_t = a_{i'})$ for symbols encoded/decoded with $\mathcal{C}^{\varnothing}$. Similarly, the vector λ_t gives the probabilities for all the symbol clock instants t^* such that $t = t^* \, mod \, L$. The SER and MSE expressions for the code $\mathcal{C}^{*,L}$ are directly given by, respectively, Eqn. 4.48 and Eqn. 4.49 with $K = L$.

Soft decoding methods also apply when a memoryless multiplexed code is periodically used. Only one slight modification of the BCJR algorithm is required. For symbol clock instants t^* where the memoryless multiplexed code is used, the probabilities $\mathbb{P}(S_{t^*} = a_i; Y_{t^*})$ are computed as

$$\mathbb{P}(S_{t^*} = a_i; Y_{t^*}) = \mathbb{P}(Y_{t^*} = y_{t^*} | S_{t^*} = a_i) \, \mathbb{P}(S_{t^*} = a_i) \tag{4.52}$$

$$= \mu_i \sum_{x \in \mathcal{C}^{\varnothing}_{a_i}} R(y_{t^*}, x), \tag{4.53}$$

instead of using the BCJR recursion formula. Note that Eqn. 4.53 is similar to the initialization of the probability $\mathbb{P}(S_1 = a_i; Y_1)$ in the BCJR algorithm.

4.8 Simulation Results

The SER and SNR performance of first-order multiplexed codes has been evaluated considering order-1 Gauss-Markov sources, with zero-mean, unit-variance and different correlation factors ($\rho = 0.5$ and $\rho = 0.9$). The source is quantized on 8 levels with a uniform quantizer. Note that this quantizer has been chosen for the sake of simplicity. The end-to-end rate/distortion performance may be further optimized by optimizing the quantization step. Notice that the reconstruction of the quantized source leads to a SNR of 13.2 dB. Note that the reconstructed symbols SNRs given here reflect only the noise induced by the channel and do not include the quantization noise. The simulations have been performed assuming BSC and AWGN channels. Unless stated otherwise, the experiments reported in this section have been performed with high priority sequences of $K_H = 1000$ symbols and the results are averaged over 1000 channel realizations. The first-order entropies of the two sources of correlation factors $\rho = 0.5$ and $\rho = 0.9$ are respectively $h^1 = 2.29$ and $h^1 = 1.52$. Their stationary entropy is $h^0 = 2.47$. The curves given here show the performance obtained for the high priority source ($\mathbf{S}_H$). The low priority source is supposed to be pre-encoded with a classical arithmetic code.

4.8.1 SER and SNR performance of memoryless and first-order multiplexed codes

The first set of experiments aimed at showing the respective performance of memoryless and first-order multiplexed codes in presence of bit errors, considering first a BSC channel. We have first started by studying the impact of the different IA strategies. Fig. 4.1 shows the theoretical SER and SNR performance obtained from Eqn. 4.20 and Eqn. 4.21 of the first-order multiplexed codes obtained for a range of bit error rates when encoding a Gauss-Markov source ($\rho = 0.5$) with different IA strategies. It can be observed that the lexicographic IA and the crossed-IA otimized with an SA algorithm outperforms the other IA methods. These two methods will be retained in the rest of the experiments.

Fig. 4.3 shows how the memoryless and first-order multiplexed codes compare with classical codes (i.e. Huffman and arithmetic codes). The memoryless codes $\mathcal{C}^{\varnothing}$ amounts to set the number of context to 1. For the first-order multiplexed codes $\mathcal{C}^{*}$, the number of contexts considered is 8. The codes are constructed from a FLC codebook with a codeword length $c = 5$ and with the two IA strategies retained : SA-based IA and crossed-IA + SA techniques. Table 4.2 gives, for a correlation factor of 0.5, both the theoretical EDL values given by Eqn. 4.3 and the EDL values obtained by simulation for several values of the codeword length c and of sequence length K_H. The EDL values given in Table 4.2 have been computed by taking into account the number of bits of the encoded sequence of low priority symbols that have really been multiplexed together with the source S_H. For a correlation factor of $\rho = 0.9$, the first-order multiplexed codes lead to an EDL of 1.70. Considering first hard decoding, Fig. 4.3 shows the strong advantage in terms of SER and SNR of first-order multiplexed codes versus arithmetic codes in presence of bit errors for similar compression efficiency. Fig. 4.3 also shows the extra gain in SER and SNR obtained when using an MMSE soft decoding technique. As expected, when the source correlation is low ($\rho \leq 0.5$), the gain brought by soft decoding techniques with respect to hard decoding is small. Similar results have been obtained with the MAP and MPM estimation criteria. However, for sources with high correlation, MPM and MMSE decoding improve respectively the SER and SNR performance.

4.8.2 Respective SER and SNR performance of multiplexed and arithmetic codes in a tandem source-channel coding scheme

A second set of experiments aimed at comparing the performance of multiplexed and arithmetic codes when used in a tandem source-channel coding chain. We consider a Gauss-Markov source with a correlation factor $\rho = 0.5$. The arithmetic coder allows in addition the exploitation of first-order source statistics. The source coders are followed by a rate $\frac{1}{2}$ recursive systematic convolutional code (RSCC) of constraint length 9, for protection and error correction. The overall rates (source+channel) of both

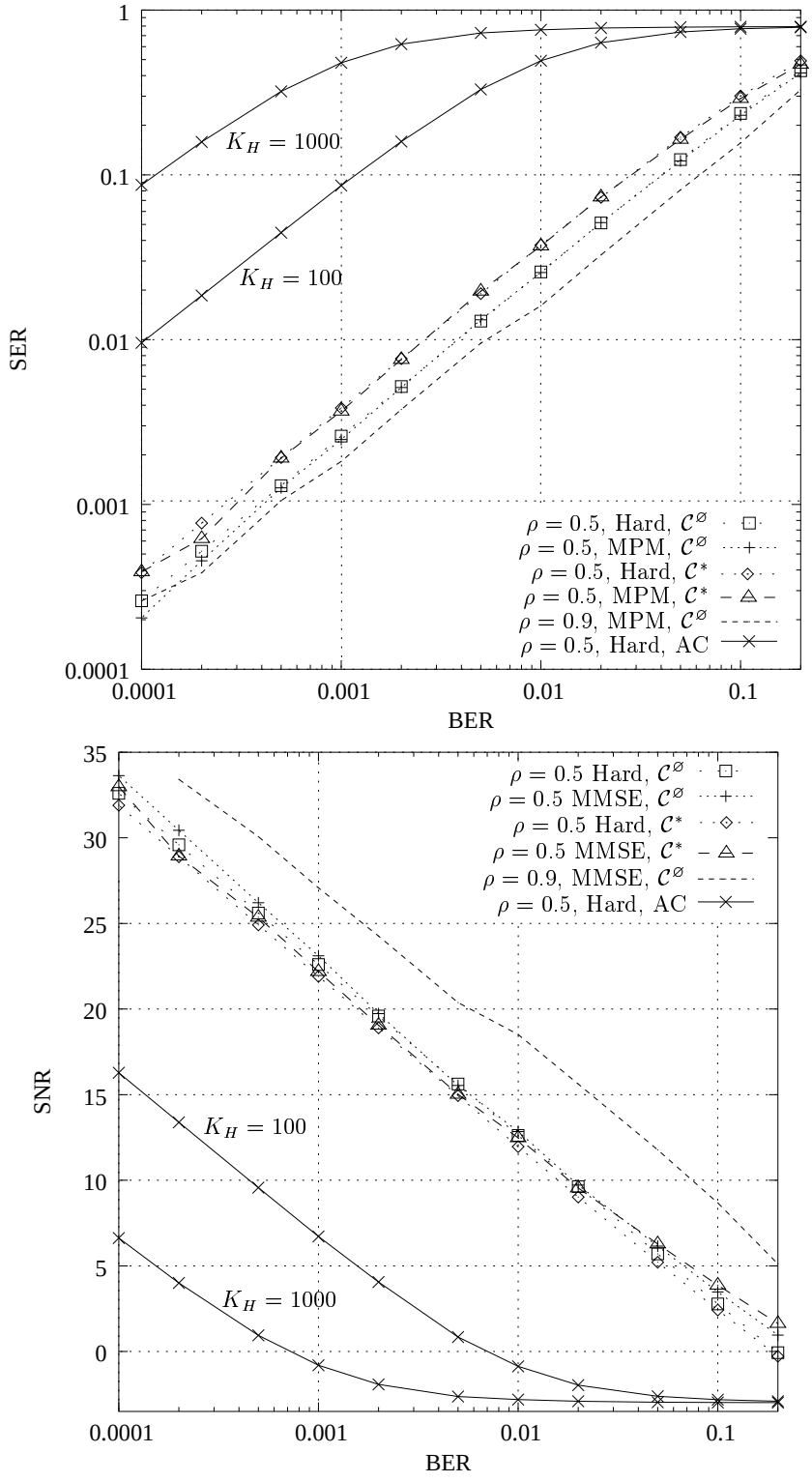


FIG. 4.3 – SER and SNR performance of memoryless and first-order multiplexed codes. The IA methods used for the two codes are based on SA and crossed-IA+SA respectively. The figure also shows the SER and SNR curves obtained with arithmetic codes (AC). Hard, MPM and MMSE decoding techniques have been considered.

chains are similar. In order to estimate the coding gain brought by the multiplexed codes, we consider an AWGN channel. To have a comparison as fair as possible, we assume that the source multiplexed codes have been designed without knowing the channel characteristics (the signal to noise ratio). In particular, a simple lexicographic IA strategy has been used. For both schemes, hard source decoding has been considered. Notice that, in both chains, soft outputs of the RSCC could be used to improve the source decoder performance.

The SER and SNR values obtained are shown in Fig. 4.4. Coding gains of about 0.75 and 1.5 dB have been obtained when using multiplexed versus arithmetic codes, for lengths of high priority sequences of $K_H = 100$ and $K_H = 1000$ respectively. The increasing gap with the sequence length results from the fact that, in contrast with arithmetic codes, multiplexed codes do not suffer from dramatic de-synchronization problems if some residual bit errors occur. This leads to significantly higher SNR and SER performance. Fig. 4.4 also shows the results obtained if the code is transmitted without channel protection. The multiplexed codes offer an inherent unequal protection of both sources without requiring the channel properties to be known. Notice that, however, when used in a classical tandem source-channel coding chain, the error correcting codes apply to both the high and the low priority source. When the two sources are encoded separately (as in the chain based on classical arithmetic codes), one has the additional freedom to use different channel codes for the two sources in the direction of unequal error protection. However, this requires the channel properties to be known at the time of encoding.

4.8.3 Synchronization of the codes

A third set of experiments aimed at evaluating the performance of the synchronization techniques based on a periodic use of memoryless multiplexed codewords. Memoryless multiplexed codewords are inserted every L symbols. The values of the parameter L vary between 2 and 50 and are shown on the curve. When L increases, the EDL obviously decreases to the value obtained when encoding the entire sequence with first-order multiplexed codes. Fig. 4.5 illustrates the corresponding EDL-SER trade-off on a BSC. The performance results shown have been obtained with MPM soft decoding. The periodic use of memoryless multiplexed codes is clearly an efficient way to decrease the SER with a low cost in terms of EDL. One can observe that the corresponding SER curve is convex and located below the dotted line curve linking the EDL-SER points of the memoryless ($\mathcal{C}^{\varnothing}$) and first-order ($\mathcal{C}^{*}$) multiplexed codes. This curve corresponds to the case where part of the sequence would be coded with memoryless codes and the rest with first-order codes in a proportion that varies along the EDL axis. The fact that the SER curve is below the dotted line curve evidences the effect of the periodic use of memoryless codes on the re-synchronization capability of the decoder.

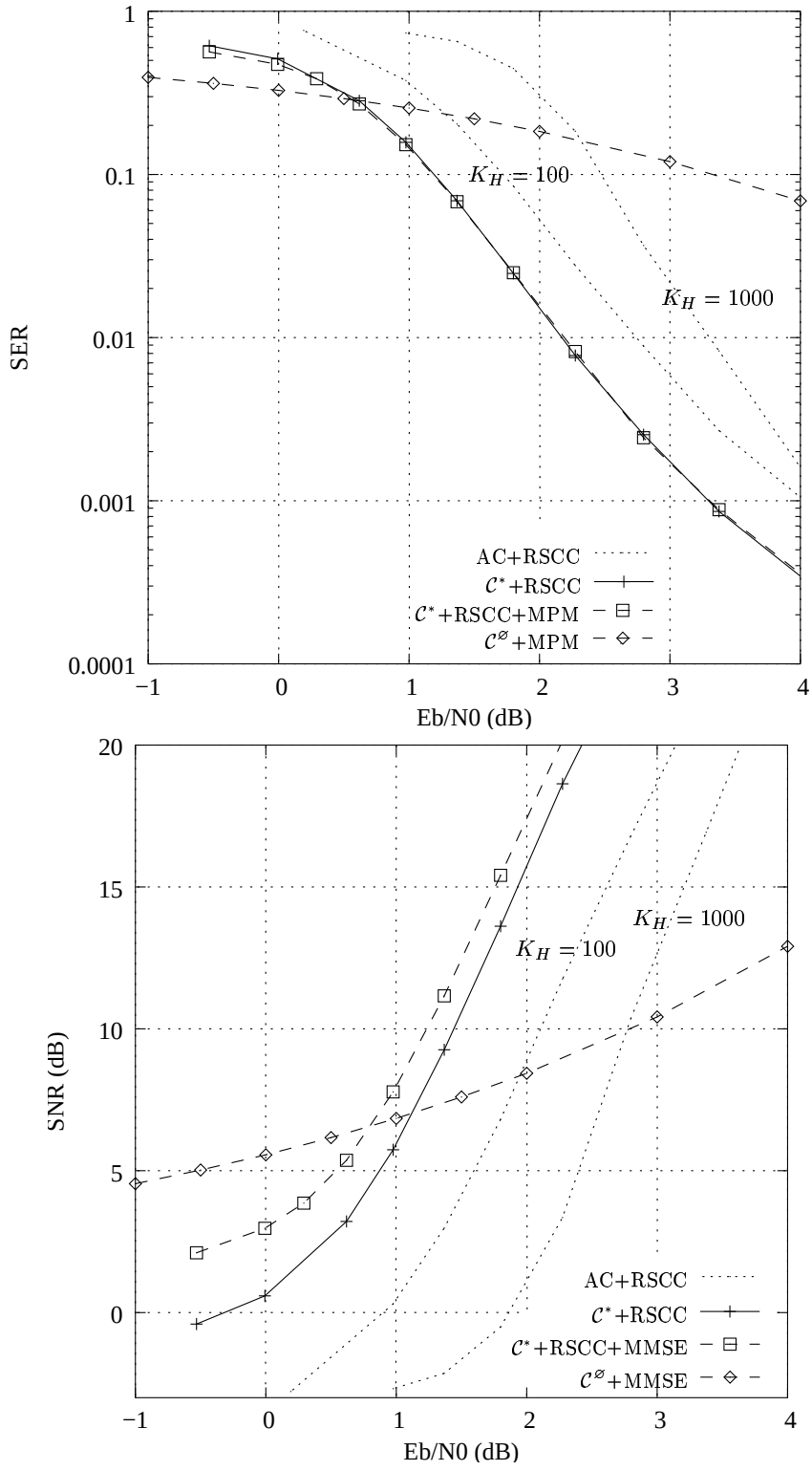


FIG. 4.4 – SER and SNR performance of multiplexed and arithmetic codes when used in a tandem source-channel coding chain. The figure shows the SER and SNR values obtained with the arithmetic codes with high priority sequences of length $K_H = 100$ and $K_H = 1000$. The multiplexed code considered relies on a lexicographic IA. The high priority sequence considered when the multiplexed code is used is $K_H = 1000$. The correlation factor of the source is $\rho = 0.5$.

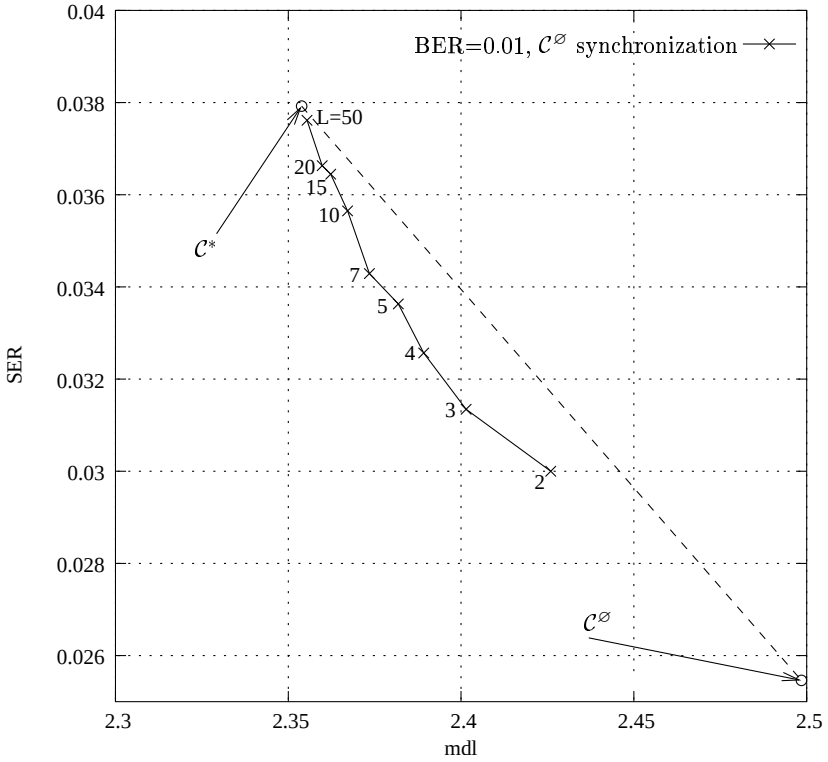


FIG. 4.5 – SER-EDL trade-off of the synchronization scheme based on a periodic use of codewords of memoryless multiplexed code every $L = 2, 3 \ldots 50$ symbols. The source considered is a Gauss-Markov source with a correlation factor of $\rho = 0.5$. The circular points are those obtained with respectively the memoryless code $\mathcal{C}^{\varnothing}$ and the first-order multiplexed code $\mathcal{C}^*$ (quasi-optimal in terms of EDL).

4.9 Conclusion

In this chapter, we have described the extension of multiplexed codes that allows the exploitation of first-order source statistics while still being error-resilient. Performance bounds in terms of SER and MSE are given. With respect to classical source codes (i.e. Huffman and arithmetic codes), the memoryless and first-order multiplexed codes are shown to have significantly higher SER and SNR performance in presence of bit errors, for the same compression efficiency. The IA having an impact on the resulting efficiency of the codes, different IA strategies have been described. It is shown in particular that a significant SER and SNR performance gain can be obtained with an IA referred to as crossed-IA that takes into account the channel characteristics. A subset of codewords, the *kernel* of the code, is also shown to have properties allowing for improved error-resilience and random data access. It is shown that, when used in a tandem source-channel coding chain, the resilience properties of the codes allow

one to obtain a coding gain up to 1.5 dB in comparison with arithmetic codes for a sequence of 1000 high priority symbols. The decoder re-synchronization capability has been shown to be further improved by a periodic use of memoryless multiplexed codewords at the expense of a controlled increased EDL.

Chapitre 5

New state models for soft decoding of variable length codes

This chapter is a joint work with Simon MALINOWSKI, *a Master student that I supervised during my PhD thesis. The work in this chapter was first presented in [JMG05b] and [JMG05a]. The contributions are shared as follows. I have introduced the aggregated trellis and proposed the study of the relationship between the error recovery of VLCs and the amenability of VLCs to integrate some termination constraints. I also proposed to use the combined trellis decoding, but most of the implementation and analysis was performed by Simon* MALINOWSKI. *The analysis of the aggregated state model and the link with error recovery properties were also addressed primarily by Simon* MALINOWSKI, *under my supervision, and submitted to [MJG05]. The last section of [MJG05], which has been addressed by Simon* MALINOWSKI *only, has* not *been inserted in this chapter (see [Mal05] for details).*

Abstract : Methodologies to analyze error recovery properties of VLCs have been introduced in [MR85] and [SD95]. In this chapter, we extend these methods to analyze the error resilience of VLCs when soft decoding with length constraint strategies are applied at the decoder side. The approach allows in particular to compute the amount of information conveyed by the length constraint on a trellis, and hence the maximum amount of information that a soft VLC decoder, augmented with a length constraint, will be able to exploit. Then, this amount of information, as well as the probability that the VLC decoder does not re-synchronize in a strict sense, are shown not to be significantly altered by appropriate trellis states aggregation. This proves that the performance of a Viterbi decoder run on aggregated state models with a length constraint can be optimal with a significantly reduced complexity compared with the bit/symbol trellis. Finally, we show that the complexity can be further decreased by projecting the state model on two state models of reduced size, and by running separate estimation.

5.1 Introduction

In recent years, many authors have considered the problem of soft decoding or joint source/channel decoding of VLCs. The approaches essentially differ in the assumptions made on the source model and on the information available at the decoder. These assumptions lead to different trellis structures on which are run the estimation or soft-decoding algorithms. Two main types of trellises are considered to estimate the sequence of emitted symbols from the received noisy bitstream : the bit-level trellis proposed in [Bal97] and the bit/symbol trellis. The bit-level trellis leads to low complexity decoding. However it does not allow one to exploit extra information, such as the number of emitted symbols, also called *termination constraint*. It hence suffers from some suboptimality. If the knowledge of the number of emitted symbols is available at the decoder, the problem is referred to as *soft decoding with length constraint* and is addressed, e.g., in [MF99][PM98][GFGR01][KT05]. This problem has lead to the introduction of the bit/symbol trellis in [BH00a]. This trellis can optimally exploit such constraints, leading to optimal performance in terms of error resilience. Nevertheless, the number of states of the bit/symbol trellis is a quadratic function of the sequence length. The corresponding complexity is actually not tractable for typical sequence lengths. In order to overcome this complexity hurdle, most authors apply suboptimal estimation methods on this optimal state model such as sequential decoding [MF98b][BKK01][KT05] or iterative decoding in the framework of bayesian networks [GFGR01].

Error recovery properties of VLCs were first studied in [MR85], where Maxted and Robinson proposed a method for computing the so-called *expected error span* E_s, i.e. the expected number of source symbols on which a single bit error propagates. For a given VLC, the lower E_s is, the better the error resilience of this code is when hard decoding is applied at the decoder side. Later, Swaszek and DiCicco have adapted in [SD95] this method to compute the so-called *synchronization gain/loss*, i.e. the probability that the number of symbols in the transmitted and decoded sequences differ by a given amount ΔS when a single bit error occurs during the transmission.

In this chapter, we show that the PMF of the synchronization gain/loss is a key feature of a VLC to analyze the error resilience of such codes when soft decoding with length constraint is applied at the decoder side. For that purpose, we first analyze in Section 5.2 the amount of information conveyed by a termination constraint on the bit/symbol trellis. It has been shown in [Wei03][TK03] that the Markovian property of a source can be easily integrated in the source model by expanding the state model by a constant factor. We thus restrict the analysis to memoryless sources. This analysis is based on the method introduced in [SD95] to compute the synchronization *gain/loss*. This method is recalled and extended to the case where a symbol sequence of length $L(S)$ is sent over a BSC of a given crossover probability. The derivation is inspired from the matricial method described in [ZZ02]. The PMF of the synchronization

gain/loss allows to compute the probability that a sequence does not resynchronise in the strict-sense as well as the maximum amount of information that a soft decoder of VLC-encoded sources, augmented with a length constraint, will be able to exploit. The latter quantity is given by the entropy of ΔS. These two quantities are shown to better predict the relative BER and Sequence Error Rate (SQER) performance of VLCs, than the mean and the variance of the error propagation length (MEPL and VEPL respectively) proposed, e.g. in [ZZ02], when soft decoding with a length constraint is applied.

This analysis is then used in Section 5.3 to show the optimality of the aggregated state models for typical source/channel realizations. The aggregated state model is defined by both the internal state of the VLC decoder (i.e., the internal node of the VLC codetree) and the rest of the Euclidean division of the symbol clock values by a fixed parameter T. This model aggregates states of the bit/symbol trellis which are distant of T instants of the symbol clock. The parameter T controls the trade-off between estimation accuracy and the decoding complexity. The choice of the parameter T has indeed an impact on the quantity of information brought by the length constraint on the corresponding trellis. Actually, we show that the probability that the VLC decoder does not re-synchronize in a strict sense, as well as the entropy of the termination constraint, are not significantly altered by aggregating states, provided that the aggregation parameter T is greater or equal than a threshold. An upper bound of this threshold is derived according to the analysis of Section 5.2. This proves that the performance of a Viterbi decoder run on the aggregated trellis can be optimal for a significantly reduced complexity.

It is also shown in Section 5.4 that the decoding complexity can be further reduced by considering separate estimation on trellises of smaller dimensions, whose parameters T_1 and T_2 are relatively prime. If the two sequence estimates are not equal, the decoding on a trellis of parameter $T_1 \times T_2$ is then computed. The equivalence between this approach, referred to as combined trellis decoding, and the decoding on a trellis of parameter $T_1 \times T_2$ is proved for the maximum a posteriori criterion, i.e. for the Viterbi algorithm [Vit67]. It is shown that the expectation of the overall computational cost is a function of the channel noise and is decreased for most values of signal to noise ratio.

5.2 Error recovery and decoding with a length constraint : a link

The transmission setup of Fig. 5.1 is considered. Let $\mathbf{S} = S_1, \ldots, S_t, \ldots, S_{L(\mathbf{S})}$ be a source of length $L(\mathbf{S})$. This source is encoded with a VLC $\mathcal{C}$, producing a bitstream $\mathbf{X} = X_1, \ldots, X_k, \ldots, X_{L(\mathbf{X})}$ of length $L(\mathbf{X})$. The bitstream $\mathbf{X}$ is modulated using a binary phase shift keying (BPSK) modulation and is transmitted over an AWGN chan-

a_i	a_1	a_2	a_3	a_4	a_5
$\mathbb{P}(a_i) =$	0.4	0.2	0.2	0.1	0.1
$\mathcal{C}_1$	00	01	10	110	111
$\mathcal{C}_2$	00	01	11	100	101
$\mathcal{C}_3$	00	10	11	010	011
$\mathcal{C}_4$	01	00	10	110	111
$\mathcal{C}_5$	01	00	11	100	101
$\mathcal{C}_6$	01	10	11	000	001
$\mathcal{C}_7$	0	10	110	1110	1111
$\mathcal{C}_8$	0	10	111	1100	1101
$\mathcal{C}_9$	0	11	100	1010	1011
$\mathcal{C}_{10}$	0	11	101	1000	1001
$\mathcal{C}_{11}$	0	100	101	110	111
$\mathcal{C}_{12}$	0	100	110	101	111
$\mathcal{C}_{13}$	0	100	111	110	101
$\mathcal{C}_{14}$	0	101	110	100	111
$\mathcal{C}_{15}$	0	101	111	100	110
$\mathcal{C}_{16}$	0	110	111	100	101

TAB. 5.1 – Source and Codes from [ZZ02] used in this chapter.

nel, without any channel protection. The channel is characterized by its signal to noise ratio, denoted E_b/N_0 and expressed in decibels. Note that we reserve capital letters to represent random variables and small letters to represent the corresponding realizations.

In this chapter, the term *polynomial* refers to expressions of the form $\sum_{i\in\mathbb{Z}} a_i x^i$, where x denotes the variable and a_i are polynomial coefficients. Hence, we include in this terminology either polynomial series (with an infinite number of non null coefficients) or finite length polynomials (such that $\exists N/\ \forall n > N,\ a_{-i} = a_i = 0$), both with negative powers.

5.2.1 The *gain/loss* behavior of a VLC

In [MR85], Maxted and Robinson propose a method to compute the so-called *expected error span* E_s following a single bit error. This method relies on an error state diagram that represents the states of the decoder when the encoder is in the root node. Hence, the error state diagram includes the internal states of the decoder, i.e. the internal nodes of the VLC, plus two states which represent the *loss of synchronization state* n_l and the *return to synchronization state* n_s, respectively. Therefore, the set of states of the diagram is $\{n_l, n_{\alpha_1}, n_{\alpha_2}, \dots, n_s\}$, where the set $\{\alpha_1, \alpha_2, \dots\}$ represents the set of strict

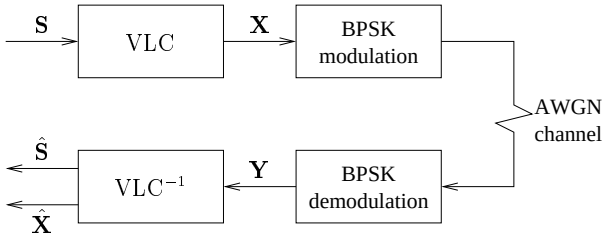


FIG. 5.1 – Transmission setup : soft decoding of a VLC with length constraint.

prefixes of the VLC, excluding the root node. The state n_s of the error state diagram corresponds to a return of both encoder and decoder automata to the root node of the code tree. However this state may not correspond to a *strict sense* synchronization. In other words, the number of decoded symbols may be different from the number of emitted symbols.

In [MR85], the branches of the error state diagram represent the transitions between two states of the decoder when a single source symbol has been emitted by the encoder. They are labelled by an indeterminate variable z which corresponds to the encoding of one source symbol. Hence the gain along each edge is the probability of the transition associated with that edge times z. In that case, the gain on the diagram from n_l to n_s (i.e. the transfer function between n_l and n_s) is a polynomial of the variable z such that the coefficient of z^i is the probability that the considered VLC resynchronizes after exactly i source symbols following the bit error. Evaluating the derivative of the gain polynomial in 1 provides the error span E_s.

Swaszek and DiCicco [SD95] have extended the branch labelling of this error state diagram so that its gain polynomial informs about the difference, caused by a single bit error, between the number of emitted and decoded symbols, after hard decoding of the received bitstream. This quantity, denoted ΔS in the sequel, is referred to as the *gain/loss*. In order to evaluate the PMF of the random variable ΔS, a new variable y^j is introduced in the branch labelling of the error state diagram. The superscript j represents the number of extra output symbols for each input symbol. Hence, the corresponding gain polynomial $G(y, z)$ is function of both variables y and z. Evaluating this polynomial for $z = 1$ gives a polynomial which is function of y only. For sake of clarity, we will simply denote this polynomial as

$$G(y) = G(y, z)|_{z=1}. \tag{5.1}$$

The coefficient of y^i in the polynomial $G(y)$ gives the probability $\mathbb{P}(\Delta S = i)$ following one bit error. Note that i can be negative if the decoded sequence is longer than the encoded one. In this section, we focus on the behavior of the polynomial

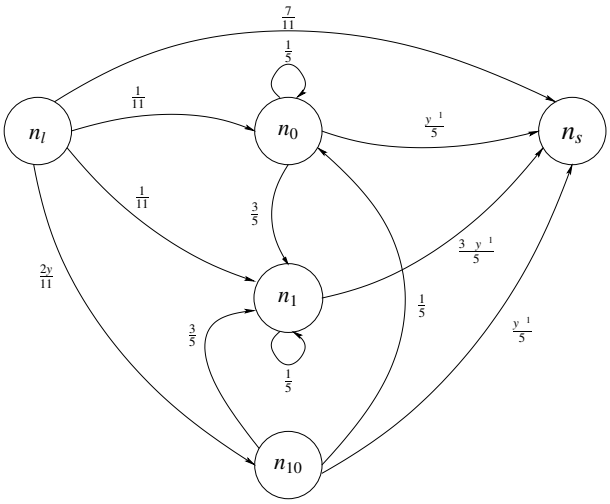


FIG. 5.2 – Error state diagram of [SD95] for the code $\mathcal{C}_5$. The transition probabilities are denoted next to the branches.

$G(y)$. Since the variable z is not necessary, we compute directly the state diagram with $z = 1$, as depicted in Fig. 5.2 for the code $\mathcal{C}_5$.

Let $\overline{H}$ be the transition matrix corresponding to the error state diagram.

$$\overline{H} = \begin{pmatrix} \mathbb{P}(n_l/n_l) & \mathbb{P}(n_{\alpha_1}/n_l) & \cdots & \mathbb{P}(n_s/n_l) \\ \mathbb{P}(n_l/n_{\alpha_1}) & \mathbb{P}(n_{\alpha_1}/n_{\alpha_1}) & \cdots & \mathbb{P}(n_s/n_{\alpha_1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}(n_l/n_s) & \mathbb{P}(n_{\alpha_1}/n_s) & \cdots & \mathbb{P}(n_s/n_s) \end{pmatrix} \tag{5.2}$$

where $\mathbb{P}(n_{\alpha_i}/n_{\alpha_j})$ represents the probability to go to state n_{α_i} from state n_{α_j}. Let us call $h^k_{i,j}(y)$ the element at row i and column j of the matrix $\overline{H}^k$. Note that $h^k_{i,j}(1)$ is the probability to go from state n_{α_i} to state n_{α_j} in k stages, i.e. after the encoding of k source symbols. The top right elements of the matrices $\overline{H}$ and $\overline{H}^k$ are respectively denoted $h(y) = \mathbb{P}(n_s/n_l)$ and $h^k(y)$. The gain polynomial $G(y)$ can then be written as

$$G(y) = \sum_{k \in \mathbb{N}^*} h^k(y). \tag{5.3}$$

Hence, this gain polynomial $G(y)$ is obtained as the top right element of the matrix $(\overline{I} - \overline{H})^{-1}$, where $\overline{I}$ denotes the identity matrix of same dimensions as $\overline{H}$. Note that this property holds if $(\overline{I} - \overline{H})^{-1}$ exists.

Example 5.1: For the code $\mathcal{C}_5$, the transition matrix derived from the previous guidelines, i.e. by setting the variable z to 1 in the extended diagram of Swaszek and

DiCicco, is given by

$$\overline{H}_{\mathcal{C}_5} = \begin{pmatrix} 0 & \frac{1}{11} & \frac{1}{11} & \frac{2y}{11} & \frac{7}{11} \\ 0 & \frac{1}{5} & \frac{3}{5} & 0 & \frac{y^{-1}}{5} \\ 0 & 0 & \frac{1}{5} & 0 & \frac{3+y^{-1}}{5} \\ 0 & \frac{1}{5} & \frac{3}{5} & 0 & \frac{y^{-1}}{5} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

This leads to $G(y) = 0.0625y^{-1} + 0.8352 + 0.1023y$*, which also means that*

$$\begin{cases} \mathbb{P}(\Delta S = -1) & = 0.1023 \\ \mathbb{P}(\Delta S = 0) & = 0.8352 \\ \mathbb{P}(\Delta S = 1) & = 0.0625. \end{cases} \tag{5.4}$$

5.2.2 Extension to the BSC

Let us recall that the previous PMF correspond to the PMF of the gain/loss engendered by a *single* bit error. We propose here to estimate $\mathbb{P}(\Delta S = i)$ for a symbol sequence of length $L(\mathbf{S})$ that has been sent through a BSC of crossover probability BER. Since in this section the VLC decoder is assumed to be a classical hard decoder, the analysis is also valid on an AWGN channel characterized by its signal to noise ratio E_b/N_0 by taking BER $= \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right)$.

The bitstream length $L(\mathbf{X})$ stands in the interval $\{L(\mathbf{S}) \times l_m, \ldots, L(\mathbf{S}) \times l_M\}$, where l_m and l_M denote the lengths of the shortest and longest codewords, respectively. For every $k \in \{L(\mathbf{S}) \times l_m, \ldots, L(\mathbf{S}) \times l_M\}$, the probability $\mathbb{P}(L(\mathbf{X}) = k)$ is calculated from the source statistics and the code $\mathcal{C}$ structure. Let E denote the random variable corresponding to the number of errors after the hard decoding of the received bitstream $\mathbf{Y}$. Its probability can be expressed as

$$\mathbb{P}(E = e) = \sum_{i \in \{L(\mathbf{S}) \times l_m, \ldots, L(\mathbf{S}) \times l_M\}} \mathbb{P}(E = e / L(\mathbf{X}) = i) \mathbb{P}(L(\mathbf{X}) = i). \tag{5.5}$$

Note that the probability $\mathbb{P}(E = e/L(\mathbf{X}) = i)$ only depends on the signal to noise ratio and is equal to

$$\mathbb{P}(E = e/L(\mathbf{X}) = i) = \binom{i}{e} \text{BER}^e \, (1 - \text{BER})^{i-e}. \tag{5.6}$$

Let us define

$$G_e(y) = \sum_{i \in \mathbb{Z}} a_{i,e} y^k, \tag{5.7}$$

where $a_{i,e} = \mathbb{P}(\Delta S = i/E = e)$. Note that the polynomial $G_1 = G$ corresponds to the gain polynomial of Eqn. 5.3.

Let us now assume that the decoder has already recovered from previous errors when another error occurs. This assumption requires that the probability that an error occurs when the decoder has not returned to the synchronization state n_s is very low and does not noticeably impact the gain polynomial G_e. Lower is the error span and higher is E_b/N_0, the more accurate is this approximation. Under this assumption, the quantity ΔS is independently impacted by multiple errors, which leads to

$$G_e(y) = (\underbrace{G \star \ldots \star G}_{e \text{ times}})(y), \tag{5.8}$$

where $\star$ denotes the convolution product. With Eqn. 5.5, the resulting gain polynomial for this BER can be expressed as

$$\tilde{G}(y) = \sum_{i \in \mathbb{Z}} \tilde{g}_i \, y^i \tag{5.9}$$

$$= \sum_{e \in \mathbb{N}} G_e(y) \, \mathbb{P}(E = e), \tag{5.10}$$

where only the quantity $\mathbb{P}(E = e)$ depends on E_b/N_0. The quantity $\mathbb{P}(\Delta S = i)$ is then given by $\tilde{g}_i$:

$$\mathbb{P}(\Delta S = i) = \tilde{g}_i = \sum_{e \in \mathbb{N}} a_{i,e} \, \mathbb{P}(E = e). \tag{5.11}$$

The gain polynomial $\tilde{G}$ is a non-finite length polynomial, but as the absolute value $|i|$ increases to infinity, $\tilde{g}_i$ vanishes to zero, due to the fact that $\mathbb{P}(E = e)$ tends to 0 as e increases (see Eqn. 5.6).

Let $\eta > 0$ be a criterion of negligibility. For a given η, the pseudo-degree d_η of the polynomial $\tilde{G}$ is defined as

$$d_\eta \triangleq \min_{\mathbb{N}^*} d \Big/ \sum_{i \in \mathbb{Z} - \{-d, \ldots, d\}} \tilde{g}_i < \eta. \tag{5.12}$$

Example 5.2: Let us determine the pseudo-degree such that $\eta = 10^{-6}$ for the code $\mathcal{C}_5$, $E_b/N_0 = 6$ dB, and $L(\mathbf{S}) = 100$. The estimates of $\tilde{g}_i$ obtained from Eqn. 5.11 lead to

$$\begin{cases} \mathbb{P}(\Delta S \leq -4) & = 0.0000002 \\ \mathbb{P}(\Delta S = -3) & = 0.0000235 \\ \mathbb{P}(\Delta S = -2) & = 0.0013201 \\ \mathbb{P}(\Delta S = -1) & = 0.0493389 \\ \mathbb{P}(\Delta S = 0) & = 0.9186664 \\ \mathbb{P}(\Delta S = 1) & = 0.0301524 \\ \mathbb{P}(\Delta S = 2) & = 0.0004930 \\ \mathbb{P}(\Delta S = 3) & = 0.0000053 \\ \mathbb{P}(\Delta S \geq 4) & = 0.0000001. \end{cases} \tag{5.13}$$

Code	d_η	$\mathbb{P}(\Delta S = 0)$	$H(\Delta S)$	MEPL[ZZ02]	VEPL[ZZ02]	NLD	BER	SQER
$\mathcal{C}_1$	3	0.9185	0.499	3.89256	34.721	0.00877	0.00193	0.34053
$\mathcal{C}_2$	4	0.9005	0.578	2.02273	2.003	0.00632	0.00191	0.33641
$\mathcal{C}_3$	4	0.8971	0.595	2.06061	2.107	0.00626	0.00192	0.33636
$\mathcal{C}_4$	4	0.8913	0.608	4.07692	27.800	0.00759	0.00177	0.31548
$\mathcal{C}_5$	3	0.9187	0.497	1.71023	1.200	0.00586	0.00194	0.34296
$\mathcal{C}_6$	4	0.8996	0.578	3.54546	18.854	0.00758	0.00182	0.32368
$\mathcal{C}_7$	5	0.7088	1.287	1.55556	0.370	0.00619	0.00154	0.21849
$\mathcal{C}_8$	10	0.7006	1.553	2.34861	2.045	0.00646	0.00134	0.19543
$\mathcal{C}_9$	9	0.6703	1.632	1.95707	1.025	0.00571	0.00123	0.16739
$\mathcal{C}_{10}$	36	0.6401	2.267	6.18182	36.231	0.00483	0.00074	0.10354
$\mathcal{C}_{11}$	8	0.8797	0.655	1.85227	2.233	0.00614	0.00183	0.32219
$\mathcal{C}_{12}$	8	0.8882	0.620	1.71678	1.506	0.00617	0.00187	0.32951
$\mathcal{C}_{13}$	8	0.8860	0.634	1.79798	1.914	0.00615	0.00182	0.32142
$\mathcal{C}_{14}$	8	0.8957	0.599	2.03104	2.952	0.00666	0.00186	0.32698
$\mathcal{C}_{15}$	8	0.8941	0.610	2.20321	4.144	0.00685	0.00189	0.33244
$\mathcal{C}_{16}$	6	0.9044	0.564	1.98086	2.615	0.00672	0.00193	0.33829

TAB. 5.2 – Pseudo-degrees d_η, proposed criteria and simulated error resilience performance for $E_b/N_0 = 6$ db and $\eta = 10^{-6}$.

Hence, according to the definition of the pseudo-degree in Eqn. 5.12, $d_\eta = 3$. *The values of* $\mathbb{P}(\Delta S = i)$ *obtained by simulation and averaged over* 10^7 *channel realizations are*

$$\begin{cases} \mathbb{P}(\Delta S \leq -4) & = 0.0000002 \\ \mathbb{P}(\Delta S = -3) & = 0.0000207 \\ \mathbb{P}(\Delta S = -2) & = 0.0012587 \\ \mathbb{P}(\Delta S = -1) & = 0.0500770 \\ \mathbb{P}(\Delta S = 0) & = 0.9185508 \\ \mathbb{P}(\Delta S = 1) & = 0.0296306 \\ \mathbb{P}(\Delta S = 2) & = 0.0004578 \\ \mathbb{P}(\Delta S = 3) & = 0.0000041 \\ \mathbb{P}(\Delta S \geq 4) & = 0.0000001 \end{cases} \tag{5.14}$$

and also lead to $d_\eta = 3$.

The simulated values of $\mathbb{P}(\Delta S = i)$ are close to the estimated ones for a large set of E_b/N_0 values, which validates the accuracy of the approximations. The pseudo-degrees of the codes introduced in Table 5.1 have been computed according to this approximation and are given in Table 5.2.

5.2.3 Code selection criteria

The previous probability mass function permits the computation of the two following quantities :

- the probability $\mathbb{P}(\Delta S = 0)$ to have a strict sense resynchronisation ;
- the entropy $H(\Delta S)$.

The optimal bit/symbol trellis, augmented with a length constraint, discards the sequences that do not satisfy the given constraint. Consequently, the probability $\mathbb{P}(\Delta S = 0)$ that the synchronization is strict sense is lower, the probability that the soft decoder detects and corrects an error is higher. Besides, the entropy $H(\Delta S)$ measures the expected amount of information conveyed to the decoder by the termination constraint. Both quantities measure the amenability of a VLC to exploit the termination constraint. Table 5.2 shows the values of these two quantities for the codes of Table 5.1, together with the MEPL and the VEPL of [ZZ02] and the decoding performance of these codes on the optimal bit/symbol trellis for E_b/N_0= 6dB.

The performance is given in terms of the normalized Levenshtein distance [Lev66] (NLD), of the BER and of the SQER. It shows that our criteria are more accurate than the ones of [ZZ02] to predict the performance of a VLC when soft decoding with length constraint is applied at the decoder side. The MEPL and VEPL criterion are indeed not designed to take into account the amount of information conveyed by the termination constraint. However, these criteria are relevant when hard decoding is applied at the decoder. On the considered set of codes, both our criteria agree that the code $\mathcal{C}_{10}$ is the best for the SQER, BER and NLD and that $\mathcal{C}_5$ is the worst code for the BER and the SQER.

5.3 State aggregation

The above analysis is used to show the optimality of the aggregated state models introduced in this chapter. This model applies when soft decoding with length constraint is applied. The optimal state model is recalled and then we introduce the state aggregation principle. It is then shown that the amount of information conveyed by the length constraint is not significantly altered by the state aggregation principle provided that the parameter T is above a given threshold.

5.3.1 Optimal state model

The whole transmission chain of Fig. 5.1 can be modeled as a hidden Markov model where the time instants represent the bit clock instants k, $1 \leq k \leq L(\mathbf{X})$. Let N_k denote the random variable corresponding to the internal state of the VLC (i.e. the internal node of the VLC codetree, which also correspond to a prefix of the code considered) at the bit clock instant k. For instance, the possible values of N_k for the code

$\mathcal{C}_0 = \{0, 10, 11\}$ are n_ε and n_1, where n_ε represents the root node of the VLC codetree. In the bit-level trellis [Bal97], the decoder state model is defined by the random variable N_k only. Consequently, the internal states of the automaton associated with a given VLC are defined by the internal nodes of the codetree, as depicted in Fig. 5.3-a for the code $\mathcal{C}_0$. The corresponding decoding trellis is given Fig. 5.4-a.

If we assume that the length of the emitted sequence is perfectly known on the decoder side, the memory of the symbol clock has to be preserved in the state model in order to optimally exploit the termination constraint. The optimal state model is defined in [BH00a] by the random variable (N_k, T_k), where T_k denotes the symbol clock instant corresponding to the bit clock instant k. Since the trellis corresponding to this model is indexed by both the bit and the symbol instants, it is often referred to as the *bit/symbol* trellis. This trellis is depicted on Fig. 5.4-b for the code $\mathcal{C}_0$.

The total number of states of the full state model is a quadratic function of the sequence length (equivalently the bitstream length). The resulting computational cost is not tractable for typical values of the sequence length $L(\mathbf{S})$.

5.3.2 Aggregated state model : a brief description

The aggregated state model is defined by the pair of random variables (N_k, M_k), where $M_k = T_k \mod T$ is the rest of the Euclidean division of T_k by T. The corresponding realization is denoted m_k. Note that $T = 1$ and $T = L(\mathbf{S})$ amounts to considering the bit-level trellis and the bit/symbol trellis, respectively. The automaton and decoding trellis of parameter $T = 2$ corresponding to this state model are depicted for the code $\mathcal{C}_0$ in Fig. 5.3-b and 5.4-c respectively.

The transitions which terminate in the state n_ε, that is corresponding to the encoding/decoding of a symbol, modify the modulo M_k as $M_k = M_{k-1}+1 \mod T$. Hence, the transition probabilities on this automaton are given by

$$\mathbb{P}(N_k = n_k, M_k = m_k | N_{k-1} = n_{k-1}, M_{k-1} = m_{k-1}) = \begin{cases} \mathbb{P}(N_k = n_k | N_{k-1} = n_{k-1}) & \text{if } n_k \neq n_\varepsilon \text{ and} \\ & m_k = m_{k-1} \\ \mathbb{P}(N_k = n_k | N_{k-1} = n_{k-1}) & \text{if } n_k = n_\varepsilon \text{ and} \\ & m_k = m_{k-1} + 1 \mod T \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

where the probabilities $\mathbb{P}(N_k = n_k | N_{k-1} = n_{k-1})$ are deduced from the source statistics. Note that the transition probabilities $\mathbb{P}(N_k | N_{k-1})$ are the ones used in the bit-level trellis.

The proposed state model allows to preserve an information on the symbol clock. In order to exploit this information, the decoder has to know the value $m_{L(\mathbf{x})} = L(\mathbf{s}) \mod T$ of the number of emitted symbols. This information corresponds to a termination constraint, as depicted in Fig. 5.4. If this value is not given by the syntax elements

of the source coding system, it has to be transmitted. The transmission cost of this value is greater than or equal to $\log_2 T$ bits. Note that the knowledge of this value has a lower cost than the one of transmitting the exact number of emitted symbols in the bit/symbol trellis. In the following, the quantity $m_{L(\mathbf{X})}$ is assumed to be known by the decoder. The estimation is performed using the Viterbi algorithm [Vit67], hence minimizing the sequence error rate (SQER). In the sequel, the error resilience will be measured according to this criterion. For the estimation, the paths that do not satisfy the appropriate boundary constraints, i.e. the paths that do not terminate in states of the form $(n_\varepsilon, m_{L(\mathbf{X})})$, are discarded.

5.3.3 Aggregated state model : analysis

According to the definition of the pseudo-degree (Eqn. 5.12) of the polynomial $\tilde{G}(y)$, the probability that ΔS belongs to the interval $\{-d_\eta, \ldots, d_\eta\}$ is equal to $1-\eta$. This leads to the following property.

Property 5.1: A value T such that $T = d_\eta$ and *a fortiori* such that $T > d_\eta$ ensures that the Viterbi algorithm run on the aggregated trellis selects, with at least probability $1-\eta$, a sequence with the correct number of symbols.

However, this property does not mean that the algorithm will offer similar results as the ones on the optimal bit/symbol trellis. To compare the aggregated state model with the optimal one, the respective amount of information conveyed by the termination constraint on both models must be quantified. These quantities are given by the entropies of the random variables $\Delta S \mod T$ and ΔS, respectively. They depend on the sequence length and E_b/N_0, which are assumed to be fixed.

For a trellis of parameter T and following the analysis of Section. 5.2.2, the quantity

$$\tilde{g}_i^T \triangleq \mathbb{P}(\Delta S \mod T = i) \tag{5.16}$$

can be computed from the quantities $\tilde{g}_i$ as

$$\tilde{g}_i^T = \sum_{j\in\mathbb{Z}} \tilde{g}_{jT+i}. \tag{5.17}$$

The entropy of the termination constraint is then given by

$$H(\Delta S \mod T) = -\sum_{i\in\{0,\ldots,T-1\}} \tilde{g}_i^T \log_2 \tilde{g}_i^T \tag{5.18}$$

$$\geq H(\Delta S) + \sum_{i\notin\{0,\ldots,T-1\}} \tilde{g}_i \log_2 \tilde{g}_i. \tag{5.19}$$

Hence, since $\Delta S \in \{-d_\eta, d_\eta\}$,

$$H(\Delta S \mod 2d_\eta + 1) \geq H(\Delta S) + \sum_{i \notin \{-d_\eta,\ldots,d_\eta\}} \tilde{g}_i \log_2 \tilde{g}_i. \tag{5.20}$$

Let us now assume that $\eta < \frac{1}{e}$. Then the function $x \mapsto x \log x$ decreases on the interval $\{0, \ldots, \eta\}$ and since $\forall i \notin \{-d_\eta, \ldots, d_\eta\}$, $\tilde{g}_i \leq \eta$, we have

$$\sum_{i \notin \{-d_\eta,\ldots,d_\eta\}} \tilde{g}_i \log_2 \tilde{g}_i \geq |\{i \notin \{-d_\eta, \ldots, d_\eta\}, \tilde{g}_i > 0\}|\, \eta \log_2 \eta, \tag{5.21}$$

where the cardinality $|\{i, \tilde{g}_i > 0\}|$ of the set of possible non-zero values for $\tilde{g}_i$ is bounded by the bitstream length $L(\mathbf{X})$. Together with $L(\mathbf{X}) \geq T$, it leads to

$$|\{i \notin \{-d_\eta, \ldots, d_\eta\}, \tilde{g}_i > 0\}| \leq L(\mathbf{X}) - 2\, d_\eta - 1. \tag{5.22}$$

Therefore, for a given η, we have the following lower and upper bounds :

$$H(\Delta S \mod 2\, d_\eta + 1) \geq H(\Delta S) + (L(\mathbf{X}) - 2\, d_\eta - 1)\, \eta \log_2 \eta \tag{5.23}$$

and

$$H(\Delta S \mod 2\, d_\eta + 1) \leq H(\Delta S), \tag{5.24}$$

hence,

$$\lim_{\eta \to 0} H(\Delta S \mod 2\, d_\eta + 1) = H(\Delta S), \tag{5.25}$$

which means that for η small enough, i.e. for $T = 2\, d_\eta + 1$ big enough, the quantity of information that may be exploited by the decoding algorithm on the proposed trellis of parameter T is the same as the one exploited on the bit/symbol trellis.

Example 5.3: Let us consider the same parameters as in Example 5.2. From Eqn. 5.23 we deduce that

$$\begin{aligned} H(\Delta S) - H(\Delta S \mod 2\, d_\eta + 1) &\leq -(100\, l_M - 5)\eta \log_2 \eta \\ &\leq 0.04900 \textit{ bits.} \end{aligned} \tag{5.26}$$

The convergence of $H(\Delta S \mod T)$ is depicted in Fig. 5.5 for some codes of Table 5.1. Similar results may be obtained the criterion $\mathbb{P}(\Delta S \mod T = 0)$. According to Section 5.2.3, the best codes are the ones with the highest values of $H(\Delta S)$. Such codes require a higher value of T to approach the optimal value $H(\Delta S)$ of the entropy of the termination constraint on the bit/symbol trellis, since the pseudo-degree of these codes is higher. Note that for the code $\mathcal{C}_{13}$, $H(\Delta S \mod 2) = 0 = H(\Delta S \mod 1)$. This

means that the performance of the soft decoding of this code on a trellis of parameter $T = 2$ will not be improved compared to the soft decoding on the bit-level trellis ($T = 1$).

The previous analysis has been validated by simulation, for sequences of $L(\mathbf{S}) = 100$ symbols. For each parameter set (VLC, E_b/N_0 and T), the SQER is measured over 10^5 channel realizations. The performance at different values of the parameter T and for the codes $\mathcal{C}_5, \mathcal{C}_7, \mathcal{C}_{10}$ and $\mathcal{C}_{13}$ is given in Table 5.3. In this table, the optimal performance of each code for the considered values of E_b/N_0 is written in *italic*. These values correspond to the performance of the soft decoding on the optimal bit/symbol trellis, and are obtained for a value of T which is considerably lower than $L(\mathbf{S})$. As predicted, the trellis of parameter $T = 2$ does not bring any improvement in terms of error resilience for the code $\mathcal{C}_{13}$ compared to the bit-level trellis. These results validate the criteria exposed in Section 5.2.3 for selecting good codes in terms of error resilience. Indeed, according to these criteria and the simulation results, the best code among the ones proposed in Table 5.1 is the code $\mathcal{C}_{10}$ and the worst is the code $\mathcal{C}_5$.

5.4 Combined trellis decoding

5.4.1 Motivation

In this section, we propose an approach allowing further reduction of the complexity of decoding without inducing any suboptimality in terms of decoding performance. The optimality of this approach is proved for the SQER criterion. This approach is motivated by the equivalence

$$\begin{aligned} & L(\mathbf{S}) \mod (T_1 \times T_2) = m \\ \Leftrightarrow & \begin{cases} L(\mathbf{S}) \mod T_1 = m \mod T_1, \\ L(\mathbf{S}) \mod T_2 = m \mod T_2, \end{cases} \end{aligned} \tag{5.27}$$

satisfied if T_1 and T_2 are relatively prime. Note that, if T_1 and T_2 are not relatively prime, the converse is not satisfied.

Property 5.2: Let us assume that T_1 and T_2 are relatively prime and that $T_3 \triangleq T_1 \times T_2$. Let us denote by $\hat{\mathbf{S}}_1$, $\hat{\mathbf{S}}_2$ and $\hat{\mathbf{S}}_3$ the estimates of $\mathbf{S}$ provided by the Viterbi algorithm run on the trellises of parameters T_1, T_2 and T_3 respectively. Then we have

$$\hat{\mathbf{S}}_1 = \hat{\mathbf{S}}_2 \Rightarrow \hat{\mathbf{S}}_3 = \hat{\mathbf{S}}_1 = \hat{\mathbf{S}}_2. \tag{5.28}$$

This means that if a sequence is selected by the trellises of parameters T_1 and T_2, then this sequence is also selected by the trellis of parameter T_3. The proof of this property is given below.

E_b/N_0	3	4	5	6	7
	Code $\mathcal{C}_5$				
$T=1$	0.99120	0.92330	0.70464	0.38774	0.14558
$T=2$	0.98805	0.90368	0.66193	0.34633	0.12452
$T=3$	0.98698	0.89901	0.65527	0.34313	0.12388
$T=4$	0.98665	0.89795	0.65457	0.34298	*0.12386*
$T=5$	0.98652	0.89782	0.65449	*0.34296*	
$T=10$	*0.98651*	*0.89780*	*0.65448*		
	Code $\mathcal{C}_7$				
$T=1$	0.99182	0.92604	0.71405	0.39372	0.14885
$T=2$	0.98634	0.88506	0.59864	0.25742	0.06997
$T=3$	0.98247	0.86379	0.55406	0.22571	0.06152
$T=4$	0.98005	0.85387	0.53964	0.21947	0.06059
$T=5$	0.97893	0.84960	0.53581	0.21866	*0.06057*
$T=10$	0.97773	*0.84731*	*0.53468*	*0.21849*	
$T=20$	*0.97772*				
	Code $\mathcal{C}_{10}$				
$T=1$	0.97993	0.87316	0.61783	0.31353	0.11390
$T=2$	0.96917	0.82122	0.51758	0.22232	0.06832
$T=3$	0.96092	0.78516	0.46126	0.18023	0.05207
$T=4$	0.95331	0.75512	0.41127	0.14437	0.03718
$T=5$	0.94755	0.73502	0.38403	0.12851	0.03226
$T=10$	0.93238	0.68744	0.33174	0.10496	0.02631
$T=20$	0.92801	0.67825	0.32560	*0.10354*	*0.02610*
$T=30$	*0.92791*	*0.67811*	*0.32558*		
	Code $\mathcal{C}_{13}$				
$T=1$	0.98973	0.91752	0.69351	0.38031	0.14431
$T=2$	0.98973	0.91752	0.69351	0.38031	0.14431
$T=3$	0.98369	0.88547	0.62816	0.32182	0.11644
$T=4$	0.98552	0.89259	0.63858	0.32711	0.11762
$T=5$	0.98286	0.88356	0.62642	*0.32142*	*0.11638*
$T=10$	0.98286	0.88356	0.62642		
$T=20$	*0.98277*	*0.88348*	*0.62638*		

TAB. 5.3 – SQER for hard decoding and soft decoding (Viterbi) with different values of the aggregation parameter T.

Proof: We emphasize that the probability of a sequence, computed by the Viterbi algorithm on a trellis of parameter T, does not depend on T. Let us assume that, if two sequences have the same probability, then a subsidiary rule is applied to select one sequence amongst the two. For instance, the lexicographical order can be chosen as a comparison rule. Such a rule ensures the Viterbi algorithm behavior to be deterministic. Let

$$\mathcal{S}_T \triangleq \{s'/L(s') \mod T = L(s) \mod T\} \tag{5.29}$$

be the set of sequences satisfying the termination constraint for the trellis of parameter T. From Eqn. 5.27, we deduce that if $T_3 = T_1 \times T_2$ with T_1 and T_2 relatively prime, then

$$\mathcal{S}_{T_3} = \mathcal{S}_{T_1} \cap \mathcal{S}_{T_2}, \tag{5.30}$$

hence,

$$\mathcal{S}_{T_3} \subseteq \mathcal{S}_{T_1}. \tag{5.31}$$

Moreover, since we have assumed that $\hat{\mathbf{S}}_1 = \hat{\mathbf{S}}_2$, we have

$$\hat{\mathbf{S}}_1 \in \mathcal{S}_{T_3}. \tag{5.32}$$

The estimate $\hat{\mathbf{S}}_1$ provided by the Viterbi algorithm applied on the trellis of parameter T_1 is then such that

$$\hat{\mathbf{S}}_1 = \arg \max_{s' \in \mathcal{S}_{T_1}} \mathbb{P}(s'|\mathbf{X}) \tag{5.33}$$

$$= \arg \max_{s' \in \mathcal{S}_{T_3}} \mathbb{P}(s'|\mathbf{X}) \tag{5.34}$$

$$= \hat{\mathbf{S}}_3, \tag{5.35}$$

where the subsidiary rule may be used in the selection of the maximum. ■

5.4.2 The decoding algorithm

The purpose of the algorithm described in this section is to exploit the property 5.2. The corresponding approach is referred to as *combined trellis decoding* henceforward. The rationale behind this approach is to use two trellises of parameters T_1 and T_2 instead of the trellis of parameter $T = T_1 \times T_2$ in order to reduce the overall decoding complexity. We will also assume that the greatest common divisor (GCD) of T_1 and T_2 is 1, i.e. that T_1 and T_2 are relatively prime. The decoding of a sequence proceeds as follows :

1. The Viterbi algorithm is applied to both trellises T_1 and T_2. Note that these two decoding algorithms may be run in parallel. They provide the estimated sequences $\hat{\mathbf{S}}_1$ and $\hat{\mathbf{S}}_2$ respectively.

2. If $\hat{\mathbf{S}}_1 = \hat{\mathbf{S}}_2$, the decoded sequence is used as the estimate of the emitted sequence.
3. Else, the Viterbi algorithm is applied to the trellis of parameter $T_1 \times T_2$.

According to Property 5.2, if the same sequence is selected by both trellises T_1 and T_2, this sequence is also selected by the trellis of parameter $T_1 \times T_2$. Consequently, we have the following property.

Property 5.3: Let $T_1, T_2 \in \mathbb{N}$ such that $\text{GCD}(T_1, T_2) = 1$. The combined trellis decoding approach of parameters T_1 and T_2 leads to the selection of the same sequence as the trellis of parameter $T_1 \times T_2$.

Hence, the proposed combined trellis decoder of parameters T_1 and T_2 is equivalent to a Viterbi decoder operating on the trellis of parameter $T_1 \times T_2$.

5.4.3 Expected computational cost of the proposed algorithm

First, let us recall that if $T = 1$, the resulting trellis is equivalent to the bit-level trellis. If T is greater or equal than $L(\mathbf{S}) - \frac{L(\mathbf{X})}{l_M} + 1$ (*a fortiori* to $L(\mathbf{S})$), the trellis is equivalent to the bit/symbol trellis. The intermediate values of T amount to consider trellises whose complexity is lower than the one of the bit/symbol trellis. Since the number of states is almost proportional to the quantity T, the computational cost D_T corresponding to the trellis of parameter T can be approximated as

$$D_T \approx T \times D_{\text{bal}}, \tag{5.36}$$

where D_{bal} denotes the computational cost of the bit-level trellis.

The expectation $D_{\text{mtd}}(T_1, T_2)$ of the computational cost of the proposed decoding scheme is then given by

$$D_{\text{mtd}}(T_1, T_2) = T_1 D_{\text{bal}} + T_2 D_{\text{bal}} + \rho T_1 T_2 D_{\text{bal}}, \tag{5.37}$$

where $\rho = \mathbb{P}(\hat{\mathbf{S}}_1 \neq \hat{\mathbf{S}}_2)$. In the following, $D_{\text{mtd}}(T_1, T_2)$ will be denoted D_{mtd}. The proposed method is worthwhile in terms of computational cost if $D_{\text{mtd}} < T_1 \times T_2 \times D_{\text{bal}}$, i.e. if

$$\rho < \rho^* = 1 - \frac{T_1 + T_2}{T_1 \times T_2}. \tag{5.38}$$

Therefore, the benefit of the proposed algorithm depends on the probability ρ that the two estimators return the same sequence estimate. Therefore, the channel noise is higher the sequence is longer and the probability that the estimators agree is lower.

Fig. 5.6 illustrates, for the code $\mathcal{C}_5$ the complexity reduction brought by the combined trellis decoding algorithm for equal decoding performance. For these particular settings, a lower computational cost is obtained with this approach as long as E_b/N_0 is greater than 0.65 dB.

5.4.4 Constrained optimization of trellis parameters T_1 and T_2

In this section, the trellis parameters T_1 and T_2 are optimized so that the expected computational cost D_{mtd} is minimized with the constraint that $T_1 \times T_2 \geq T_c$, where T_c is the parameter that corresponds to a targeted decoding performance. It amounts to solving

$$\min_{T_1,T_2 \ / \ T_1T_2 \geq T_c} T_1 + T_2 + \rho T_1 T_2. \tag{5.39}$$

Note that, since here $L(\mathbf{S})$ and E_b/N_0 are assumed to be constant, the quantity $\rho = \mathbb{P}(\hat{S}_1 = \hat{S}_2)$ is fixed. Let us first assume that the tuple (T_1, T_2) belongs to $\mathbb{R}^+ \times \mathbb{R}^+$. The Lagrangian optimization of Eqn. 5.39 leads to $T_1 = T_2 = \sqrt{T_c}$. Together with Eqn. 5.36, we deduce that D_{mtd} satisfies

$$D_{\text{mtd}} \geq 2D_{\sqrt{T_c}} + \rho D_{T_c}. \tag{5.40}$$

However, this lower bound is not tight. Indeed, it is obtained only if the equality $T_1 = T_2 = \sqrt{T_c}$ holds and can subsequently not be reached since

- $T_1, T_2 \in \mathbb{N}^*$,
- T_1 and T_2 must be relatively prime.

Fig. 5.7 illustrates the set of points $(T_1 \times T_2, D_{\text{mtd}})$ that corresponds to valid tuples (T_1, T_2) when $\rho = 0.1$.

In order to be close to the optimum (i.e. $T_1 = T_2$) and to satisfy the previous constraints, a natural choice for T_1 and T_2 is $T_2 = T_1 + 1$, with $T_1 \in \mathbb{N}^*$. Since two consecutive positive integers are always relatively prime, such a choice is always valid. We are now going to show that such a choice can not be outperformed for any T_c of the form $T_c = T(T+1),\ T \in \mathbb{N}^*$.

Without loss of generality, let us assume that $T_2 = T_1 + \Delta T$, and $T_c = T_1 \times (T_1 + \Delta T)$. Note that parsing the set $\mathbb{N}^* \times \mathbb{N}^*$ with the tuple $(T_1, \Delta T)$ ensures to parse the set of attainable constraints. The corresponding computational cost D_{mtd} is given by

$$\begin{aligned} D_{\text{mtd}} &= \rho D_{T_c} + D_{\text{bal}}(2\,T_1 + \Delta T) && (5.41)\\ &= \rho D_{T_c} + 2D_{\text{bal}}\sqrt{T_c + \frac{\Delta T^2}{4}} && (5.42)\\ &\leq \rho D_{T_c} + 2\sqrt{T_c} D_{\text{bal}} + \Delta T D_{\text{bal}}. && (5.43) \end{aligned}$$

From Eqn. 5.42, we straightforwardly deduce the following property.

Property 5.4: Let $T_c \in \mathbb{N}^*$ and $\mathcal{R}_p \subseteq N^* \times N^*$ be the subset of positive integers which are relatively prime. Then

$$\arg \min_{T_1,T_2 \in \mathcal{R}_p \ / \ T_1T_2 = T_c} D_{\text{mtd}} = \arg \min_{T_1,T_2 \in \mathcal{R}_p \ / \ T_1T_2 = T_c} |T_2 - T_1|. \tag{5.44}$$

Since $\Delta T \geq 1$, from Eqn. 5.42 one can refine the lower bound of Eqn. 5.40 as

$$D_{\text{mtd}} \geq \rho D_{T_c} + 2D_{\text{bal}}\sqrt{T_c + \frac{1}{4}}, \tag{5.45}$$

where the equality stands if and only if $\Delta T = 1$. In that sense, the set of tuples (T_1, T_2) such that $T_2 = T_1 + 1$ is optimum. This condition is depicted on Fig. 5.7.

5.4.5 Extension of the combined trellis decoding algorithm

It is shown above that the combined decoding algorithm of parameters T_1 and T_2 is equivalent to the trellis of parameter $T_1 \times T_2$. Now, let us modify the step 3 of the combined decoding algorithm so that the Viterbi algorithm is run on a trellis of parameter T_c instead of $T_1 \times T_2$, where T_c is such that

$$T_1 + T_2 + \rho\, T_c < T_c \leq T_1 \times T_2. \tag{5.46}$$

In that case, the property 5.3 does not strictly hold anymore. However it is quite reasonable to assume that the performance obtained in this modified version of the combined trellis decoding is similar to the ones obtained with the trellis of parameter T_c. The optimization problem of Eqn. 5.39 is then simplified and amounts to solving

$$\min_{T_1,T_2 \,/\, \begin{cases} T_1 T_2 \geq T_c \\ \text{GCD}(T_1, T_2) = 1 \end{cases}} T_1 + T_2. \tag{5.47}$$

It can be easily shown than the minimum obtained in Eqn. 5.39 is then lower or equal to the minimum obtained in Eqn. 5.47, i.e. allows to reach the targetted performance level T_c with a lower complexity.

5.5 Conclusion

We have shown in this chapter how the synchronization properties of the VLCs reflect the error resilience of these codes when a soft decoding with length constraint is applied. These properties allow the proof of the optimality of the trellis state aggregation proposed in this chapter, and hence to reach the optimal performance of soft decoding of VLCs with a much lower decoding complexity than the one obtained on the optimal bit/symbol trellis. This complexity is further reduced by the combined trellis algorithm without inducing any suboptimality in terms of error resilience performance. The aggregated trellises can easily be coupled with a convolutional code or a turbo-code in an iterative structure, as it is done in [GFGR01], [BH00c] for the bit/symbol trellis and in [JV04] for the bit-level trellis.

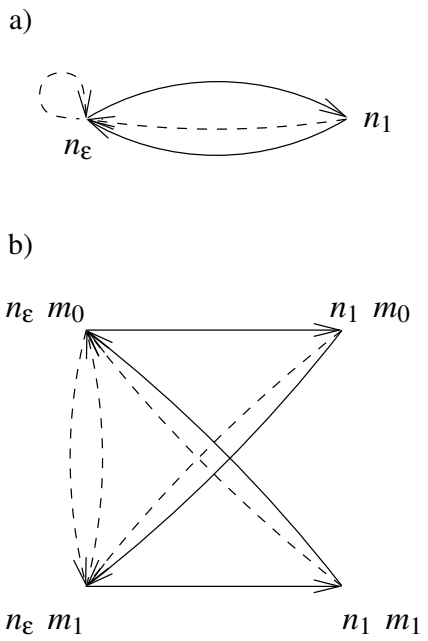


FIG. 5.3 – Automata of the state models for a) $T = 1$, b) $T = 2$ corresponding respectively to the bit-level trellis and the extended trellis with $T = 2$ (Code $\mathcal{C}_0$).

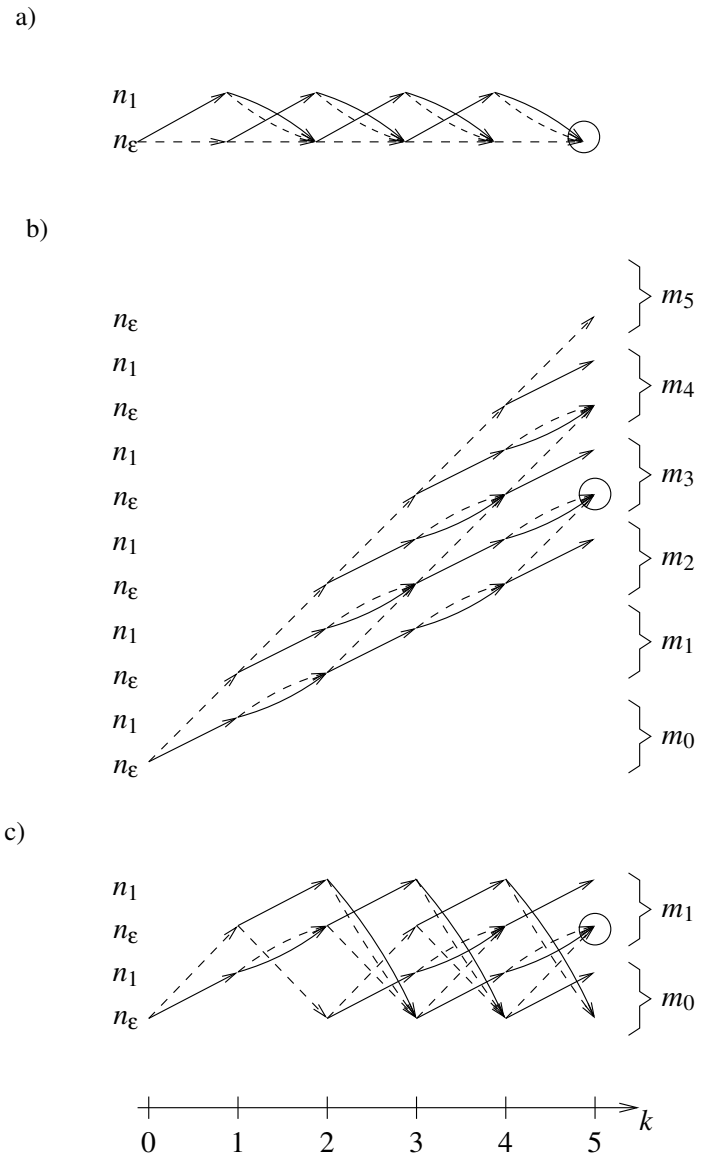


FIG. 5.4 – Trellises for the code $\mathcal{C}_0$: a) bit-level trellis ($T = 1$), b) bit/symbol trellis and c) trellis of parameter $T = 2$. The termination constraints are also depicted (here $L(\mathbf{S})$ is assumed to be odd).

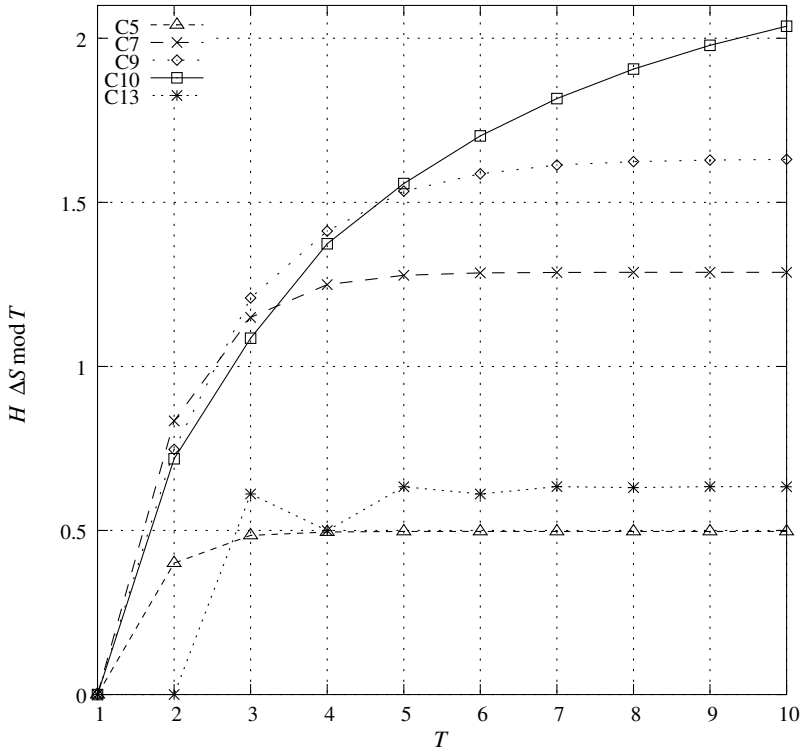


FIG. 5.5 – Entropy of $\Delta S \mod T$ versus T for the codes $\mathcal{C}_5, \mathcal{C}_7, \mathcal{C}_9, \mathcal{C}_{10}$ and $\mathcal{C}_{13}$.

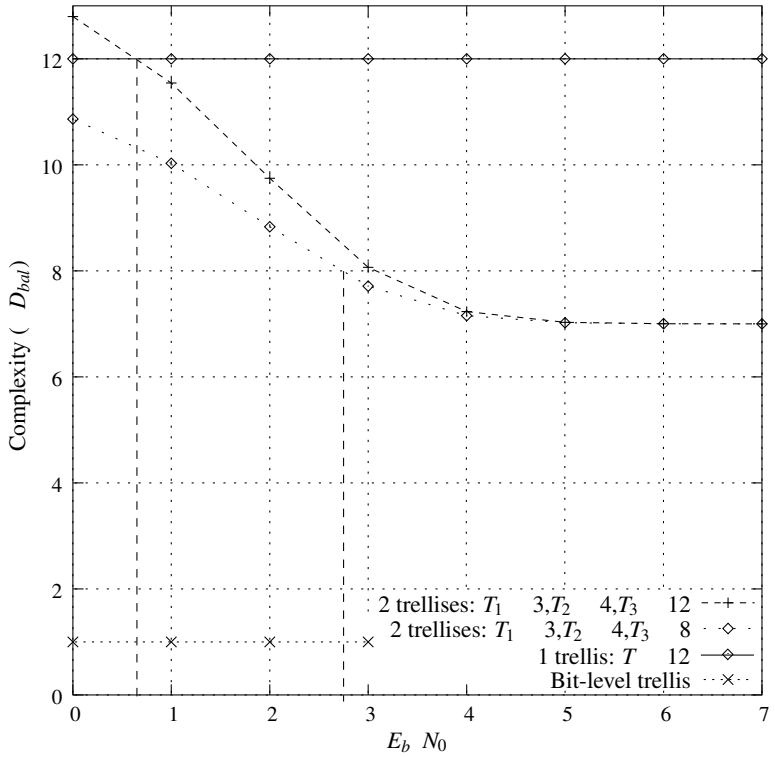


FIG. 5.6 – Computational cost of the combined trellis decoding approach versus E_b/N_0 against the computational cost of a single trellis decoding approach for parameters $(T_1 = 3, T_2 = 4, T_3 = 12)$ and for the code $\mathcal{C}_5$. The corresponding cut-off value of E_b/N_0 is also depicted and is obtained for $\rho^* = 0.417$.

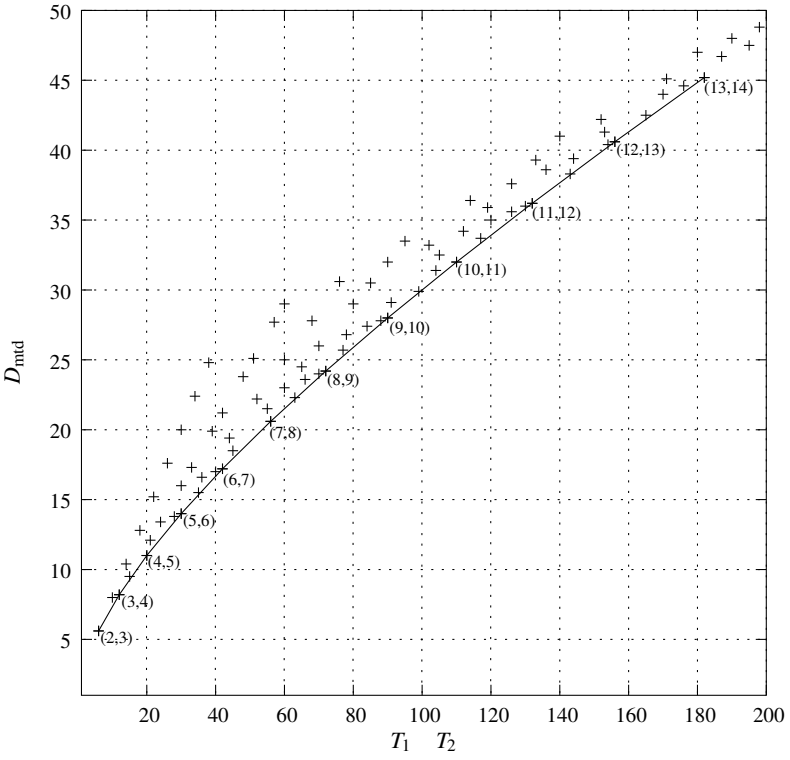


FIG. 5.7 – Computational cost of the combined trellis decoding approach versus $T_1 \times T_2$ for $\rho = 0.1$. The points plotted are such that GCD$(T_1, T_2) = 1$ and $T_1, T_2 \leq 20$.

Chapitre 6

Bitstream construction strategies for VLCs

This chapter has been submitted and accepted for publication in [JG05c]. It has been presented in part in [JG04a] and is related to the work of [JG04c] (not included in this document).

Abstract : This chapter addresses the issue of robust and progressive transmission of signals (e.g., images, video) encoded with VLCs over error-prone channels. This chapter first describes bitstream construction methods offering good properties in terms of error-resilience and progressivity. In contrast with related algorithms described in the literature, all the proposed methods have a linear complexity as the sequence length increases. The applicability of soft-input soft-output (SISO) and turbo decoding principles to resulting bitstream structures is investigated. In addition to error-resilience, the amenability of the bitstream construction methods to progressive decoding is considered. The problem of code design for achieving good performance in terms of error-resilience and progressive decoding with these transmission strategies is then addressed. The VLC code has to be such that the symbol energy *is mainly concentrated on the first bits of the symbol representation (i.e. on the first transitions of the corresponding codetree). Simulation results reveal high performance in terms of SER and MSE. These error-resilience and progressivity properties are obtained without any penalty in compression efficiency. Codes with such properties are of strong interest for the binarization of M-ary sources in state-of-the-art image and video coding systems making use of e.g., the EBCOT, or CABAC algorithms. A prior statistical analysis of the signal allows the construction of the appropriate binarization code.*

6.1 Introduction

For a given number of source symbols, the number of bits produced by a VLC coder is a random variable. The soft decoding of these codes consists in properly segmenting the noisy bitstream into measures on symbols and in estimating the symbols

from the noisy sequence of bits (or measurements) that is received. This segmentation problem can be addressed by introducing a priori information in the bitstream, taking often the form of synchronization patterns. This a priori information is then exploited to formulate constraints. One can alternatively, by properly structuring the bitstream, reveal and exploit constraints on some bit positions [RK96]. A structure of fixed length size slots inherently creates hard synchronization points in the bitstream. The resulting bitstream structure is called EREC. The principle can however be pushed further in order to optimize criteria of resilience, computational complexity and progressivity.

In this chapter, given a VLC, we first focus on the design of transmission schemes of the codewords in order to achieve high SER and SNR performance in presence of transmission errors. The process for constructing the bitstream is regarded as a dynamic bit mapping between an intermediate binary representation of the sequence of symbols and the bitstream to be transmitted on the channel. The intermediate representation is obtained by assigning codewords to the different symbols. The decoder proceeds similarly with a bit mapping which, in presence of transmission noise, may not be the inverse of the mapping realized on the sender side, leading to potential decoder de-synchronization. The mapping can also be regarded as the construction of a new VLC for the entire sequence of symbols. Maximum error-resilience is achieved when the highest number of bit mappings (performed by coder and decoder) are deterministic. *Constant* and *stable* mappings with different synchronization properties in presence of transmission errors are introduced. This general framework leads naturally to several versions of the transmission scheme, exploiting the different mapping properties. By contrast with the EREC algorithm, all the proposed algorithms have a linear complexity as the sequence length increases. The bitstream construction methods presented leads to significant improvements in terms of SER and SNR with respect to classical transmission schemes where the variable length codewords are simply concatenated. The proposed approach may be coupled with other complementary approaches of the literature. In particular, granted that the channel properties are known, UEP schemes using RCPC [Hag88] or specific approaches such as [CP05] may be used to improve the error-resilience.

Another design criterion that we consider is the amenability of VLCs and of transmission schemes for progressive decoding. The notion of progressive decoding is very important for image, video and audio applications. This is among the features that have been targeted in the embedded stream representation existing in the JPEG2000 standard. For this purpose, an expectation-based decoding procedure is introduced. In order to obtain best progressive SNR performance in presence of transmission errors, the VLC codetrees have to be designed in such a way that most of the symbols *energy* is concentrated on transitions on the codetree corresponding to bits that will be mapped in a deterministic way. Given a VLC tree (e.g. a Huffman codetree [Huf52]), one can build a new codetree, by re-assigning the codewords to the different symbols in order

to satisfy at best the above criterion, while maintaining the same EDL for the corresponding source. This leads to codes referred to as *pseudo-lexicographic* codes. The lexicographic order can also be enforced by the mean of the Hu-Tucker algorithm [HT71]. This algorithm returns the lexicographic code having the smallest EDL. Among potential applications of these codes, one can cite the error-resilient transmission of images and videos, or the binarization step of coding algorithms such as EBCOT [Tau99] and CABAC [MSW03] used in state-of-the-art image and video coders/decoders. A prior analysis of the statistical distributions of the signals to be encoded (e.g., wavelet coefficients, residue signals) allows the design of binarization codes with appropriate properties of energy compaction on the first transitions of the codetree. This in turn leads to higher MSE decrease when decoding the first bit-planes (or bins) transmitted.

The rest of the chapter is organized as follows. Section 6.2 introduces the framework of bitstream construction, the notations and definitions used. Several bitstream construction methods offering different trade-offs in terms of error-resilience and complexity are described in Section 6.3. The application of the SISO and the turbo decoding principles to the bitstream resulting from a constant mapping is described in Section 6.4. The code design is discussed in Section 6.5 and some choices are advocated. Simulation results are provided and discussed in section 6.6. The performance of the bitstream construction methods and of the codes is also assessed with a simple image coder. The amenability of VLCs to be used as a binarization tool for modern video coders is discussed.

6.2 Problem statement and definitions

Let $\mathbf{S} = (S_1, \ldots, S_t, \ldots, S_K)$ be a sequence of source symbols taking their values in a finite alphabet $\mathcal{A}$ composed of $|\mathcal{A}|$ symbols, $\mathcal{A} = \{a_1, \ldots, a_i, \ldots, a_{|\mathcal{A}|}\}$. These symbols can be wavelet or other transform coefficients which have been quantized. Let $\mathcal{C}$ be a binary variable length code designed for this alphabet, according to its stationary probability $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_i, \ldots, \mu_{|\mathcal{A}|}\}$. To each symbol S_t is associated a codeword $\mathcal{C}(S_t) = B_t^1 \ldots B_t^{L(S_t)}$ of length $L(S_t)$. The sequence of symbols $\mathbf{S}$ is converted into an intermediate representation,

$$\mathbf{B} = (\mathbf{B_1}; \ldots; \mathbf{B_K}), \tag{6.1}$$

where $\mathbf{B_t}$ is a column vector defined as

$$\mathbf{B_t} = \begin{pmatrix} B_t^1 \\ \vdots \\ B_t^{L(S_t)} \end{pmatrix}. \tag{6.2}$$

In what follows, the emitted bitstream is denoted $\mathbf{E} = E_1 \ldots E_{K_E}$ and the received sequence of noisy bits is denoted $\hat{\mathbf{E}} = \hat{E}_1 \ldots \hat{E}_{K_E}$. Similarly, the intermediate representation on the receiver side is referred to as $\hat{\mathbf{B}}$.

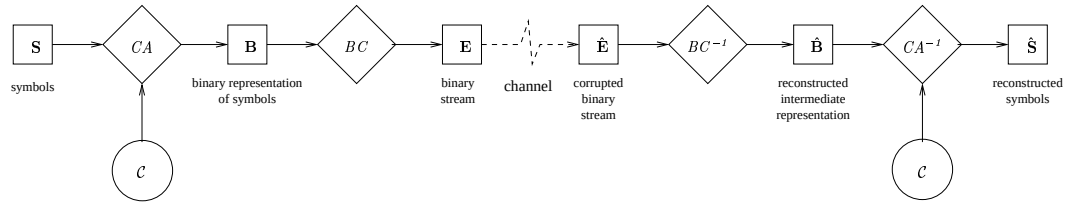


FIG. 6.1 – Coding and decoding building blocks with the code $\mathcal{C}$: codeword assignment and bitstream construction.

We consider the general framework depicted in Fig. 6.1, where the coding process is decomposed into two steps : codeword assignment (CA) and bitstream construction (BC). In classical compression systems, the codewords produced are transmitted *sequentially*, forming a *concatenated* bitstream. Here, we focus on the problem of designing algorithms for constructing bitstreams that will satisfy various properties of resiliency and progressivity. Note that both the sequence length K and the length $K_E = \sum_{t=1}^{K} L(S_t)$ of the constructed bitstream $\mathbf{E}$ are assumed to be known on the decoder side. Note also that we reserve capital letters to represent random variables. Small letters will be used to denote the values or realizations of these variables.

The BC algorithms can be regarded as dynamic bit *mappings* between the intermediate representation $\mathbf{B}$ and the bitstream $\mathbf{E}$. These mappings φ are thus defined on the set $\mathcal{I}(\mathbf{b}) = \{(t,l) \ / \ 1 \leq t \leq K,\ 1 \leq l \leq L(s_t)\}$ of tuples (t,l) that parses $\mathbf{b}$ (the realization of $\mathbf{B}$) as

$$\begin{array}{ccc} \mathcal{I}(\mathbf{b}) & \rightarrow & \{1, \cdots, K_E\} \\ (t,l) & \mapsto & \varphi(t,l) = n, \end{array} \tag{6.3}$$

where n stands for a bit position of $\mathbf{E}$. Note that the index l can be regarded as the index of a layer (or a bit-plane) in the coded representation of the symbol. Similarly, the decoder proceeds with a bit mapping between the received bitstream $\hat{\mathbf{E}}$ and an intermediate representation $\hat{\mathbf{B}}$ of the received sequence of codewords. This mapping, referred to as ψ, depends on the noisy realization $\hat{\mathbf{b}}$ of $\hat{\mathbf{B}}$ and is defined as

$$\begin{array}{ccc} \{1, \cdots, K_E\} & \rightarrow & \mathcal{I}(\hat{\mathbf{b}}) \\ n & \mapsto & \psi(n) = (t,l), \end{array} \tag{6.4}$$

where the set $\mathcal{I}(\hat{\mathbf{b}})$, in presence of bit errors, may not be equal to $\mathcal{I}(\mathbf{b})$. The composed function $\pi = \psi \, o \, \varphi$ is a dynamic mapping function from $\mathcal{I}(\mathbf{b})$ into $\mathcal{I}(\hat{\mathbf{b}})$. An element is decoded in the correct position if and only if $\pi(t,l) = (t,l)$. The error-resilience depends on the capability, in presence of channel errors, to map a bitstream element n of $\hat{\mathbf{E}}$ to the correct position (t,l) in the intermediate representation $\hat{\mathbf{B}}$ on the receiver side.

Définition 6.1 *An element index* (t,l) *is said to be* constant *by* $\pi = \psi \, o \, \varphi$*, iff* $n = \varphi(t,l)$ *does not depend on the realization* $\mathbf{b}$*. Similarly, the bitstream index* n *is also said to be* constant.

Let $\mathcal{I}_C$ denote the set of *constant* indexes. The restriction φ_C of φ to the definition set $\mathcal{I}_C$ and its inverse $\psi_C = \varphi_C^{-1}$ are also said to be *constant*. Such constant mappings can not be altered by channel noise : $\forall \hat{\mathbf{b}},\ (t,l) \in \mathcal{I}_C \Rightarrow \pi(t,l) = (t,l)$. Let l_m and l_M denote the length of the shortest and of the longest codewords of the codetree, respectively.

Définition 6.2 *An element index* (t,l) *is said to be* stable *by* π *iff* $\varphi(t,l)$ *only depends on* $B_t^1, \cdots, B_t^{l-1}$ *and* $\forall\, l'/1 \leq l' < l,\ (t,l')$ *is* stable.

Let $\mathcal{I}_S$ denote the set of *stable* indexes. A stable mapping φ_S can be defined by restricting the mapping φ to the definition set $\mathcal{I}_S$. For the set of stable indexes, the error propagation is restricted to the symbol itself.

Let us consider the transmission of a VLC encoded source on a Binary Symmetric Channel (BSC) with a BER p. Provided that there is no inter-symbol dependence, the probability that a symbol S_t is correctly decoded is given by $\mathbb{P}(\hat{S}_t = s_t | S_t = s_t) = (1-p)^{L(S_t)}$, leading to the following SER bound

$$\begin{aligned} \mathrm{SER}_{\mathrm{bound}}(\mathcal{C}) &= 1 - \sum_{a_i \in \mathcal{A}} \mu_i (1-p)^{L(a_i)} \\ &= p\, h_{\mathcal{C}} + \mathcal{O}(p^2), \end{aligned} \tag{6.5}$$

where $h_{\mathcal{C}}$ denotes the EDL of the code $\mathcal{C}$. This equation provides a lower bound in terms of SER when transmitting sources encoded with the code $\mathcal{C}$ on a BSC, assuming that simple hard decoding is used. Note that this bound is lower than the SER that would be achieved with FLCs.

6.3 Bitstream construction algorithms

In this section, we describe practical bitstream construction algorithms offering different trade-offs in terms of error-resilience and complexity.

6.3.1 Constant mapping (CMA)

Given a code $\mathcal{C}$, the first approach maximizes the cardinality of the definition set of the constant mapping φ_C, i.e. such that $\mathcal{I}_C = \{1, \cdots, K\} \times \{1, \cdots, l_m\}$. Notice first that a variable length codetree comprises a section of a fixed length equal to the minimum length of a codeword denoted l_m, followed by a variable length section. A *constant* mapping can thus be defined as the composition of functions φ_C : $\{1, \cdots, K\} \times \{1, \cdots, l_m\} \rightarrow \{1, \cdots, K\, l_m\}$ and $\psi_C = \varphi_C^{-1}$ defined such that

$$(t,l) \rightarrow \varphi_C(t,l) = (l-1)K + t \tag{6.6}$$

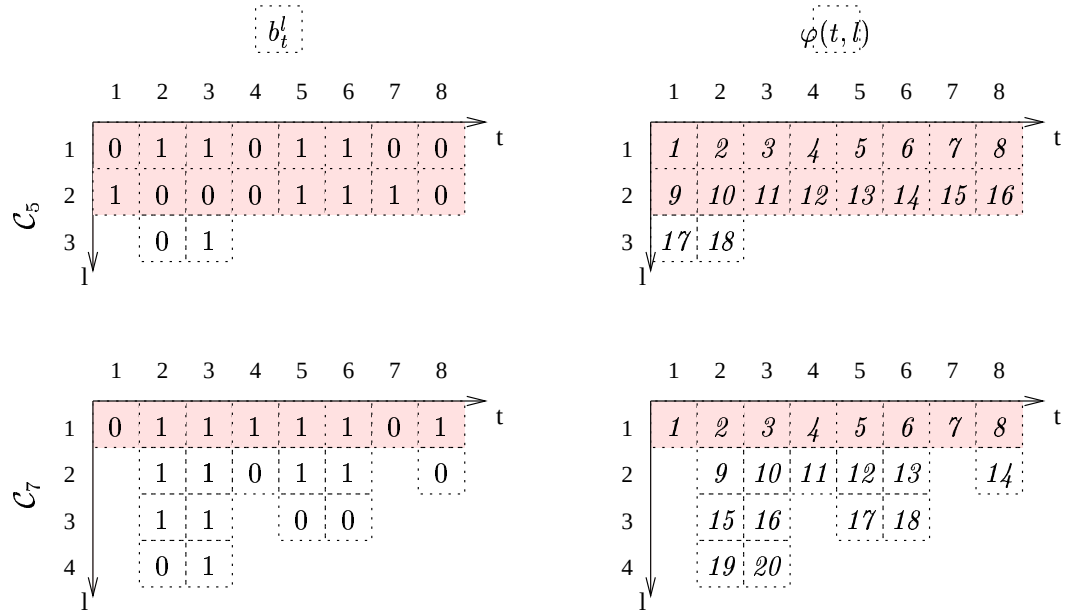


FIG. 6.2 – Example of intermediate representation $\mathbf{b}$ and corresponding mapping φ realized by the CMA algorithm. The set $\mathcal{I}_C$ of constant element indexes is highlighted in gray.

The bits that do not belong to the definition set of φ_C can be simply concatenated at the end of the bitstream. The *constant* mapping φ_C defines a set of "hard" synchronization points.

Example 6.1: Let $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$ be the alphabet of the source $\mathbf{S}_{(1)}$ with the stationary probabilities given by $\mu_1 = 0.4$, $\mu_2 = 0.2$, $\mu_3 = 0.2$, $\mu_4 = 0.1$ and $\mu_5 = 0.1$. This source has been considered by several authors, e.g. in [MR85] and [ZZ02]. The codes referred to as $\mathcal{C}_5 = \{01, 00, 11, 100, 101\}$ and $\mathcal{C}_7 = \{0, 10, 110, 1110, 1111\}$ in [ZZ02] are considered here. The realization $\mathbf{s} = a_1a_4a_5a_2a_3a_3a_1a_2$ leads to the sequence length $K_E = 18$ for code $\mathcal{C}_5$ and to the sequence length $K_E = 20$ for code $\mathcal{C}_7$ respectively. The respective intermediate representations associated with the sequence of symbols $\mathbf{s}$ are given in Fig. 6.2. The CMA algorithm proceeds with the mapping φ of the elements (b_t^l) to, respectively, positions $n = 1 \dots 18$ and $n = 1 \dots 20$ of the bitstream, as illustrated in Fig. 6.2. This finally leads to the bitstreams $\mathbf{e}$=01101100100011100 1 and $\mathbf{e}$=011111101 110111010100 for $\mathcal{C}_5$ and $\mathcal{C}_7$ respectively. Note that the set $\mathcal{I}_C$ of constant indexes associated with $\mathcal{C}_5$ and $\mathcal{C}_7$ is $\{1, \cdots, 8\} \times \{1, 2\}$ and $\{1, \cdots, 8\} \times \{1\}$, respectively.

Error propagation will only take place on the tuples (t, l) which do not belong to $\mathcal{I}_C$. The above mapping means transmitting the fixed length section of the codewords

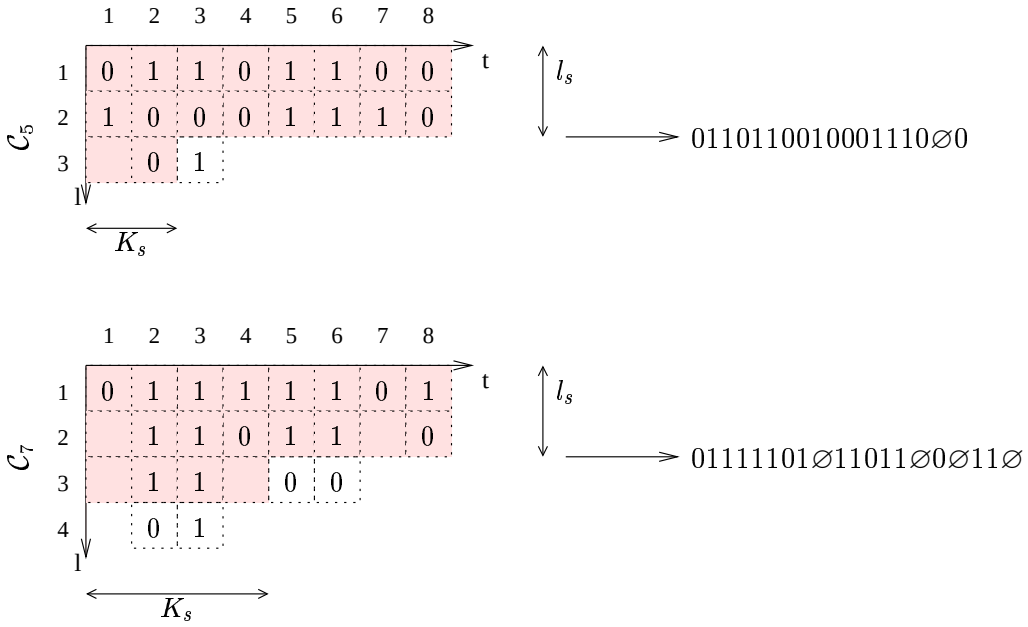


FIG. 6.3 – Definition set $\mathcal{I}_S$ (elements in gray) of the stable mapping (SMA algorithm).

bit-plane per bit-plane. Hence, for a Huffman tree, the most frequent symbols will not suffer from de-synchronization.

6.3.2 Stable mapping (SMA) algorithm

The CMA algorithm maximizes the cardinality of the definition set $\mathcal{I}_C$. The error-resilience can be further increased by trying to maximize the number of *stable positions*, i.e., by minimizing the number of inter-symbol dependencies. The stability property can be guaranteed for a set $\mathcal{I}_S$ of element indexes (t, l) defined as $\mathcal{I}_S = \{(t, l) \in \mathcal{I}(\mathbf{b})/1 \leq t \leq K, 1 \leq l \leq l_s\} \cup \{(t, l) \in \mathcal{I}(\mathbf{b})/1 \leq t \leq K_s, l = l_s + 1\}$ for l_s and K_s satisfying $l_s \times K + K_s \leq K_E$. The expectation of $|\mathcal{I}_S|$ will be maximized by choosing $l_s = \lfloor \frac{K_E}{K} \rfloor$ and $K_s = K_E \mod K$. Let us remind that $\mathcal{I}(\mathbf{b})$ is the definition set of the realization $\mathbf{b}$ of the intermediate representation $\mathbf{B}$. The set $\mathcal{I}_S$ can be seen as the restriction of $\mathcal{I}_F = (\{1, \cdots, K\} \times \{1, \cdots, l_s\}) \cup (\{1, \cdots, K_s\} \times \{l_s + 1\})$, definition set of a mapping independent of $\mathbf{b}$, to $\mathcal{I}(\mathbf{b})$. Note that $|\mathcal{I}_F| = K_E$. On the sender side, the approach is thus straightforward and is illustrated in the example below. The decoder, knowing the values of the parameters l_s and K_s, can similarly compute the restriction of $\mathcal{I}_F$ to $\mathcal{I}(\hat{\mathbf{b}})$ instead of $\mathcal{I}(\mathbf{b})$.

Example 6.2: Considering the source $\mathbf{S}_{(1)}$*, the codes and the sequence of symbols of Ex. 6.1, the SMA* algorithm *leads to the mapping of the stable indexes depicted in*

Fig. 6.3. The notation ∅ stands for bitstream positions that have not been mapped during this stage. Remaining elements of the intermediate representation, here respectively 1 *and* 0001 *for* $\mathcal{C}_5$ *and* $\mathcal{C}_7$*, are inserted in positions identified by the valuation ∅. This leads to the bitstreams* $\mathbf{e} = 01101100\,10001110\,10$ *and* $\mathbf{e} = 01111101\,01101100\,0111$ *for* $\mathcal{C}_5$ *and* $\mathcal{C}_7$ *respectively.*

At that stage some analogies with the first step of the EREC algorithm [RK96] can be shown. The EREC algorithm structures the bitstream into M slots, the goal being to create hard synchronization points at the beginning of each slot. The EREC algorithm thus leads to the creation of a constant mapping on a definition set $|\mathcal{I}_C| = M\,l_m \leq K\,l_m$. Hence, it appears that, for a number of slots lower than K (the number of symbols), the number of bits mapped in a constant manner is not maximized. This suggests to use a constant mapping on the definition set $\{1, \cdots, K\} \times \{1, \cdots, l_m\}$ and to apply EREC on slots for the remaining bits to be mapped. The corresponding algorithm is called CMA-EREC in what follows. Note that, if M=K, CMA-EREC is identical to EREC applied on a symbol basis (which satisfies $|\mathcal{I}_C| = K\,l_m$). If $M = 1$, CMA-EREC is identical to CMA. The choice of M has a direct impact on the trade-off between resilience and complexity.

6.3.3 Stable smpping construction relying on a stack-based algorithm (SMA-stack)

This section describes an alternative to the above stable mapping algorithms offering the advantage of having a linear complexity ($\mathcal{O}(K)$) with respect to EREC. Let us consider two stacks, $Stack_b$ and $Stack_p$, dedicated to store bit values b_t^l of $\mathbf{b}$ and bitstream positions n of $\mathbf{e}$ respectively. These stacks are void when the algorithm starts. Let us consider a structure of the bitstream $\mathbf{e}$ in M slots with $M = K$, i.e., with one slot per symbol s_t (the slots will be also indexed by t). The size of slot t is denoted m_t. There are K_s slots such that $m_t = l_s$ and $K - K_s$ slots such that $m_t = l_s + 1$. For each slot t, the algorithm proceeds as follows :

1. the first $\min(m_t, L(s_t))$ bits of the codeword $\mathbf{b}_t$ associated with the symbol s_t are placed sequentially in slot t.
2. if $L(s_t) > m_t$, the remaining bits of $\mathbf{b}_t$ (i.e. $b_t^{m_{t+1}} \ldots b_t^{L(s_t)}$) are put in the *reverse* order on the top of the stack $Stack_b$.
3. Otherwise, if $L(s_t) < m_t$, some positions of slot t remain unused. These positions are inserted on the top of the stack $Stack_p$.
4. While both stacks are not void, the top bit of $Stack_b$ is retrieved and inserted in the position of $\mathbf{e}$ indexed by the position that is on the top of the position stack $Stack_p$. Both stacks are updated.

After the last step, i.e. once the slot K has been processed, both stacks are void. The decoder proceeds similarly by storing (respectively retrieving) bits in a stack $Stack_b$

depending on the respective values of the codeword lengths $L(s_t)$ and of the slot size m_t. By construction of the slot structure, the number of stable elements is the same for both the SMA and the SMA-stack algorithm. The main difference between these algorithms resides in the way the remaining elements are mapped. Using the proposed stack-based procedure increases the error-resilience of the corresponding bits.

Example 6.3: Let us consider again the source $\mathbf{S}_{(1)}$ of Ex. 6.1 with the code $\mathcal{C}_7$. Fig. 6.4 illustrates how the SMA-stack algorithm proceeds. In this example, the slot structure has been chosen so that each slot is formed of contiguous bit positions, but this is not mandatory.

Note that the overhead introduced by this BC method is only due to the extra information on the sequence length K. In the VLC soft decoding litterature, his information is often assumed to be known.

6.3.4 Layered bitstream

The previous BC algorithms decrease the impact of the error propagation induced by the channel errors. Another interesting feature is the amenability of the BC framework for progressive decoding. In section 6.4.3, we will see that the different bit transitions of a binary codetree convey different amount of energy. In a context of progressive decoding, the bits which will lead to the highest decrease in reconstruction error should be transmitted first. This idea underlies the principle of bit-plane coding in standard compression solutions. The approach considered here is however more general.

The approach consists in transmitting the bits according to a given order $\succ$. To each bit to be transmitted one can assign an internal node of the codetree. One can thus relate the transmission order of the bits of the different codewords to a mapping of the internal nodes of the codetree into the bitstream. Thus, if $n_i \succ n_j$, all the bits corresponding to the internal node n_i are transmitted before any of the bits corresponding to the internal node n_j. This order induces a partition of the transitions on the codetree into segments of same "priorities". The bits corresponding to a segment of a given priority in the codetree are mapped sequentially. Note that this order may not be a total order : some transitions corresponding to distinct internal nodes may belong to the same segment. The order $\succ$ must satisfy the rule

$$n_j \not\succ n_i \text{ iff } n_j \in \mathcal{L}_i, \tag{6.7}$$

where $\mathcal{L}_i$ denotes the leaves attached to the node n_i in the binary codetree corresponding to the VLC code. This rule is required because of the causality relationship between nodes n_i and n_j.

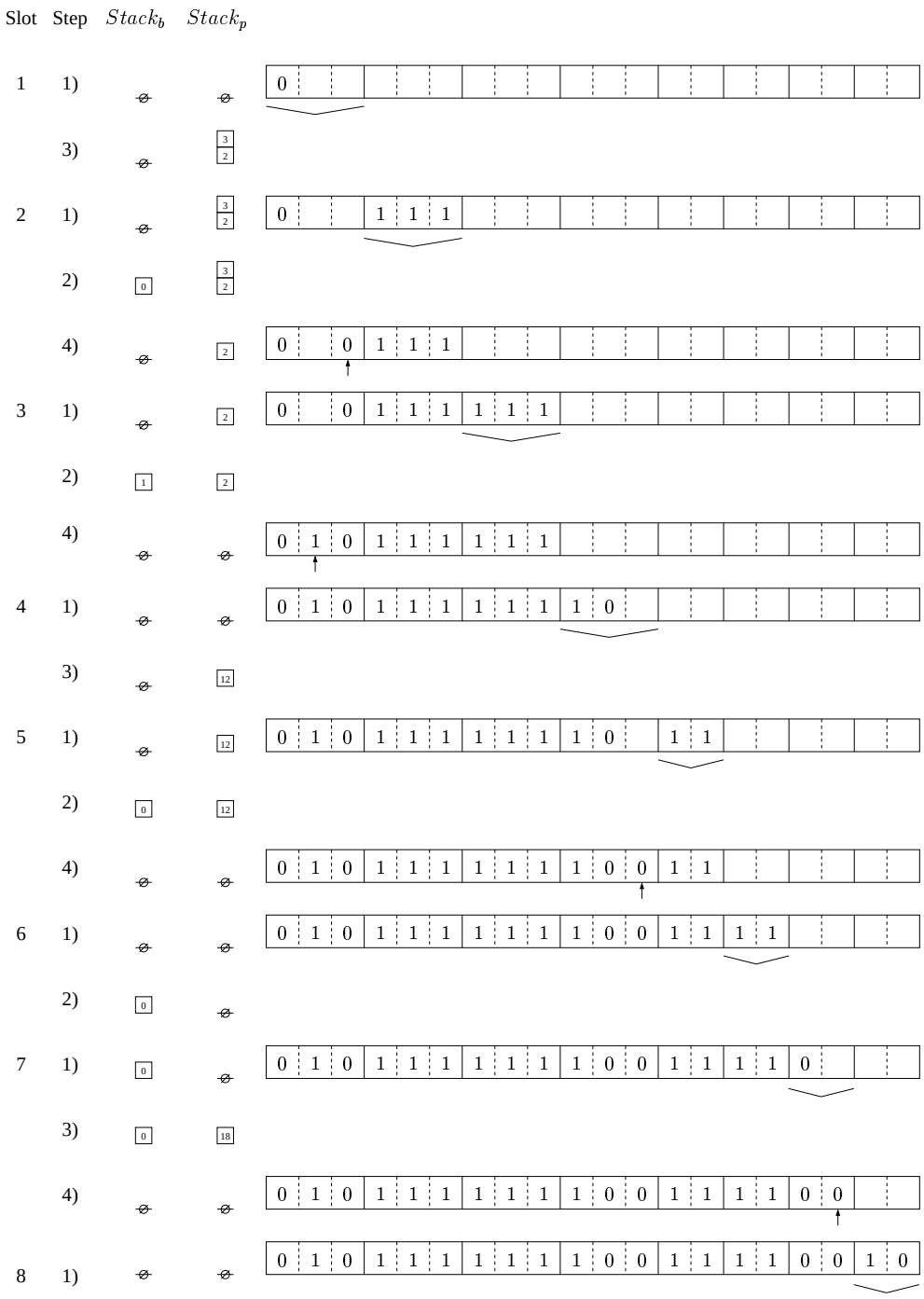


FIG. 6.4 – *Example 6.6.3 :* Encoding of sequence $a_1a_4a_5a_2a_3a_3a_1a_2$ using code $\mathcal{C}_7$ and algorithm SMA-stack.

Example 6.4: Let us consider again code $\mathcal{C}_5$. There are four internal nodes : the root /, 0, 1 and 10. These nodes are respectively referred to as n_0, n_1, n_2, n_3. A strict bit-plane [1] approach corresponds to the order $n_0 \succ n_1, n_2 \succ n_3$. Here, nodes n_1 and n_2 are mapped in the same segment. This order ensures that all the bits corresponding to a given "bit-plane" are transmitted before any of the bits corresponding to a deeper "bit-plane". Using this order, the realization $\mathbf{s} = a_1a_4a_5a_2a_3a_3a_1a_2$ of Ex. 6.1 is coded into the following bitstream :

$$\underbrace{01101100}_{n_0}\underbrace{1000110}_{n_1 \text{ and } n_2}\underbrace{01}_{n_3}.$$

Another possible order is the order $n_0 \succ n_2 \succ n_3 \succ n_1$. In that case, since the order is total, segments are composed of homogeneous bit transitions, i.e. bit transitions corresponding to the same internal node in the codetree. Then, the realization $\mathbf{s}$ is transmitted as

$$\underbrace{01101100}_{n_0}\underbrace{0011}_{n_2}\underbrace{01}_{n_3}\underbrace{1010}_{n_1}.$$

Note that the CMA algorithm leads to a bitstream with a layered structure defined by the order $n_0 \succ n_1, n_2, n_3$. Note also that the concatenation of codewords correspond to the less restrictive order between internal nodes, i.e. $\forall (n_i, n_j), n_i \not\succ n_j$.

The layered construction bitstream is an efficient way of enhancing the performance of UEP schemes. Most authors have considered the UEP problem from the channel rates point of view, i.e., by finding the set of channel rates leading to the lowest overall distortion. For this purpose, RCPC [Hag88] are generally used. The UEP problem is then regarded as an optimization problem taking as an input the relative source importance for the reconstruction. Here, the UEP problem is seen from the source point of view. It is summarized by the following question : how the bitstream construction impacts the localization of energy so that UEP methods may apply efficiently ? Since the bit segments have a different impact on the reconstruction, these segments act as sources of various importance. In the usual framework, no distinction is performed among concatenated bits corresponding to different internal nodes. Using the layered approach, one can differentiate the bits according to their impact on the reconstruction and subsequently applies the appropriate protection. UEP techniques can thus be applied in a straightforward manner. The codetree itself has clearly an impact on the energy repartition, and has to be optimized in terms of the amount of source reconstruction energy conveyed by the different transitions on the codetree. The code design is discussed in Section 6.5.

[1] The bit-plane approach for VLCs is somewhat similar to the one defined in the CABAC [MSW03].

6.4 Decoding algorithms

6.4.1 Applying the soft decoding principles (CMA algorithm)

Trellis-based soft-decision decoding techniques making use of Bayesian estimators can be used to further improve the decoding SER and SNR performance. Assuming that the sequence $\mathbf{S}$ can be modeled as a Markov process, MAP, MPM or MMSE estimators, using e.g. the BCJR algorithm [BCJR74], can be run on the trellis representation of the source model [BH00d]. This section describes how the BCJR algorithm can be applied for decoding a bitstream resulting from a constant mapping (CMA algorithm).

Let us consider a symbol-clock trellis representation of the product model of the Markov source with the coder model [GFGR01]. For a given symbol index (or symbol clock instant) t, the state variable on the trellis is defined by the pair (S_t, N'_t) where N'_t denotes the number of bits *used to encode the first t symbols*. The value n'_t taken by the random variable N'_t is thus given by $n'_t = \sum_{t'=1}^{t} l(s_{t'})$. Notice that, in the case of classical transmission schemes where the codewords are simply concatenated, n'_t=n, where n is the current bit position in the bitstream $\mathbf{e}$.

The BCJR algorithm computes the probabilities $\mathbb{P}(S_t = a_i|\hat{e}_1; \ldots; \hat{e}_{K_E})$ knowing the Markov source transitions probabilities $\mathbb{P}(S_t = a_i|S_{t-1} = a_{i'})$, and the channel transition probabilities $\mathbb{P}(\hat{E}_n = \hat{e}_n|E_n = e_n)$, assumed to follow a DMC model. Using similar notations to that of [BCJR74], the estimation proceeds with forward and backward recursive computations of the quantities

$$\alpha_t(a_i, n'_t) = \mathbb{P}(S_t = a_i; N'_t = n'_t; (\hat{e}_{\varphi(t',l)})), \tag{6.8}$$

where $(\hat{e}_{\varphi(t',l)})$ denotes the sequence of received bits in bitstream positions $n = \varphi(t', l)$, with $1 \leq t' \leq t$ and $1 \leq l \leq L(s_{t'})$, and

$$\beta_t(a_i, n'_t) = \mathbb{P}((\hat{e}_{\varphi(t',l)})|S_t = a_i; N'_t = n'_t), \tag{6.9}$$

where $(\hat{e}_{\varphi(t',l)})$ denotes the sequence of received bits in bitstream positions $n = \varphi(t', l)$, with $t + 1 \leq t' \leq K$ and $1 \leq l \leq L(s_{t'})$. The recursive computation of the quantities $\alpha_t(a_i, n'_t)$ and $\beta_t(a_i, n'_t)$ requires to calculate

$$\begin{aligned} \gamma_t(a_j, n'_{t-1}, a_i, n'_t) &= \mathbb{P}(S_t = a_i; N'_t = n'_t; (\hat{e}_{\varphi(t,l)})_{1\leq l\leq L(a_i)}|S_{t-1} = a_j; N'_{t-1} = n'_{t-1}) \\ &= \delta_{n_t - n_{t-1} - L(a_i)}\, \mathbb{P}(S_t = a_i|S_{t-1} = a_j) \prod_{l=1}^{L(a_i)} \mathbb{P}(\hat{e}_{\varphi(t,l)}|b_t^l). \end{aligned} \tag{6.10}$$

In the case of a simple concatenation of codewords, $\varphi(t, l) = n'_{t-1} + l$. When using a constant mapping, we have

$$\begin{aligned} \varphi(t, l) &= (l - 1)\, K + t && \text{if } l \leq l_m \\ &= (K - t)\, l_m + n'_{t-1} + l && \text{otherwise} \end{aligned} \tag{6.11}$$

The product $\lambda_t(a_i, n') = \alpha_t(a_i, n')\,\beta_t(a_i, n')$ leads naturally to the posterior marginals $\mathbb{P}(S_t, N'_t|\hat{e}_1; \ldots; \hat{e}_{K_E})$ and in turn to the MPM and MMSE estimates of the symbols S_t.

For the CMA algorithm, information on the bit and the symbol clock values is needed to compute the entities $\gamma_t(a_j, n'_{t-1}, a_i, n'_t)$. This condition is satisfied by the bit/symbol trellis [BH00b]. However, this property is not satisfied by the trellis proposed in [Bal97].

6.4.2 Turbo-decoding

The soft-decoding approach described above can be used in a joint source-channel turbo structure. For this purpose, extrinsic information must be computed on bits. This means computing the bit marginal probability $\mathbb{P}(e_t = 0 \vee 1|\hat{e}_1; \ldots; \hat{e}_{K_E})$ instead of the symbol marginal probability. The SISO VLC then acts as the inner code, the outer code being a RSCC. In the last iteration only, the symbol per symbol output distribution $\mathbb{P}(S_t = a_i|\hat{e}_1; \ldots; \hat{e}_{K_E})$ is estimated.

6.4.3 MMSE progressive decoding

The above approach reduces the SER. However, it does not take into account MSE performance in a context of progressive decoding. Progressive decoding of VLC can be realized by considering an expectation-based approach as follows. Notice that VLC codewords can be decoded progressively by regarding the bit generated by the transitions at a given level of the codetree as a bit-plane or a layer.

Let us assume that the l first bits of a codeword have been received without error. They correspond to an internal node n_j of the codetree. Let $\mathcal{L}_j$ and $\tilde{\mu_j} = \sum_{n_i \in \mathcal{L}_j} \mu_i$ respectively denote the leaves obtained from n_j and the probability associated with the node n_j. Then the optimal (i.e. with minimum MSE) reconstruction value $\tilde{a}_j$ is given by

$$\tilde{a}_j = \frac{1}{\tilde{\mu_j}} \sum_{n_i \in \mathcal{L}_j} \mu_i\, a_i. \tag{6.12}$$

The corresponding mean square error (MSE), referred to as Δ_j, is given by the variance of the source knowing the first bits, i.e., by

$$\Delta_j = \frac{1}{\tilde{\mu_j}} \sum_{a_i \in \mathcal{L}_j} \mu_i\, (a_i - \tilde{a}_j)^2. \tag{6.13}$$

Let us consider the codetree modeling the decoding process. The reception of one bit will trigger the transition from a parent node n_j to children nodes $n_{j'}$ and $n_{j''}$ depending on the bit realization. The corresponding reconstruction MSE is then decreased as $\Delta_j - \Delta_{j'}$ or $\Delta_j - \Delta_{j''}$ depending on the value of the bit received. Given a

node n_j, the expectation δ_j of the MSE decrease for the corresponding transition T_j is given by

$$\delta_j = \Delta_j - \frac{\tilde{\mu}_{j'}\,\Delta_{j'} + \tilde{\mu}_{j''}\,\Delta_{j''}}{\tilde{\mu}_{j'} + \tilde{\mu}_{j''}}. \tag{6.14}$$

The term δ_j can be seen as an amount of signal *energy*. If all the bits are used for the reconstruction, the MSE equals 0, which leads to $\mathrm{var}(\mathbf{S}) = \Delta_{\mathrm{root}} = \sum_{n_j} \tilde{\mu_j}\,\delta_j$, which can also be deduced from Eqn. 6.14. The total amount δ_l^* of reconstruction *energy* corresponding to a given layer l of a VLC codetree can then be calculated as the weighted sum of *energies* given by transitions corresponding to the given layer :

$$\delta_l^* = \sum_{T_j \text{ in layer l}} \tilde{\mu}_j\,\delta_j. \tag{6.15}$$

Remark : Note that the MMSE estimation can be further improved by applying a BCJR algorithm on the truncated bitstream, setting the transitions on the trellis that correspond to the non-received bits to their posterior marginals or to an approximated value of $\frac{1}{2}$.

Note also that, if the quantities K and K_E are both known on the receiver side, error propagation can be detected if the termination constraints are not satisfied. Here, by termination constraints, we mean that the K_E bits of $\mathbf{E}$ must lead to the decoding of K symbols of $\mathbf{S}$. In the case where the termination constraint is not satisfied, it may be better to restrict the expectation-based decoding to the bits that cannot be desynchronized (i.e., bits mapped with a constant or stable mapping).

6.5 Code design

Based on our discussion in the previous section, the code should hence be optimized in order to satisfy at best the following criteria :

1. In order to maximize the SNR performance in presence of transmission errors, the code $\mathcal{C}$ should be such that it concentrates most of the *energy* on the bits (or codetree transitions) that will not suffer from de-synchronization. In particular, if the bits that concentrate most of the energy correspond to the first bit transition of the binary codetree, the concept of *most significant* bits can also apply for VLC codewords.
2. Similarly, the progressivity depends on the amount of energy transported by the first transmitted bits. That is why the code design should be such that few bits gather most of the reconstruction energy, and these bits should be transmitted first. For this purpose, we will assume that the layered approach proposed in section 6.3.4 will be used.

3. Finally, a better energy concentration enhances the performance of the UEP techniques : since these techniques exploit the fact that different sources (here segments of bits) have various priorities, the code should be designed to enhance the heterogeneity of the bit transitions in terms of reconstruction energy.

In this section, we will regard the code optimization problem as a simplified problem consisting in maximizing the values δ_l^* for the first codetree transitions. This problem can be addressed by optimizing Eqn. 6.15, either with the Binary Switching Algorithm [ZG90] or with a simulated annealing algorithm (e.g., [Far90]). The optimization has to be processed jointly for every layer. Hence, this multi-criteria optimization requires that some weights are provided to each layer l of Eqn. 6.15. The weights associated with bit transition depend on the application. In the following, we propose two simplified approaches, led by the consideration that the lexicographic order, separating the smaller values from the greater values, -in general- concentrates most of the energy in the first layers. Obviously, a lexicographic order is relevant only for a scalar alphabet. The first approach consists in finding an optimal code (in the Huffman sense) that aims at satisfying a lexicographic order. The second approach consists in using Hu-Tucker [HT71] codes to enforce the lexicographic order.

6.5.1 Pseudo-lexicographic (p-lex) codes

Let us consider a classical VLC (e.g. using a Huffman code), associating a codeword of length l_i to each symbol a_i. One can re-assign the different symbols a_i of the source alphabet to the VLC codewords in order to try to best satisfy the above criteria. In this part, the re-assignment is performed under the constraint that the lengths of the codewords associated with the different symbols are not affected, in order to preserve the compression performance of the code. A new codetree, referred to as a *pseudo-lexicographic* (*p-lex*) VLC codetree, can be constructed as follows. Starting with the layer $l = l_M$, the nodes (including leaves) of depth l in the codetree are sorted according to their expectation value given in Eqn. 6.12. Pairs of nodes are grouped according to the resulting order. The expectation values corresponding to the parent nodes (at depth $l-1$) are in turn computed. The procedure continues until the codetree is fully constructed. Grouping together nodes having close expectation values in general contributes to increase the energy or information carried on the first transitions on the codetree.

Example 6.5: Let us consider a Gaussian source of zero-mean and standard deviation 1*, uniformly quantized on 8 cells partitioning the interval* $(-3, +3)$*. The subsequent discrete source is referred to as* $\mathbf{S}_{(2)}$ *in what follows. Probabilities and reconstruction values associated with source* $\mathbf{S}_{(2)}$ *are given by*

$$\mathcal{A} = \{-2.5112, -1.7914, -1.0738, -0.3578, 0.3578, 1.0738, 1.7914, 2.5112\}$$

Huffman node	$\mathbb{P}(n_j)$	$\tilde{a_j}$	δ_j	*p-lex* node	$\mathbb{P}(n_j)$	$\tilde{a_j}$	δ_j	Hu-Tucker node	$\mathbb{P}(n_j)$	$\tilde{a_j}$	δ_j
∅	1	0.000	0.022	∅	1	0.000	0.297	∅	1	0.000	0.626
0	0.434	+0.170	0.477	0	0.566	−0.4776	0.013	0	0.5	−0.791	0.228
1	0.566	−0.131	0.224	1	0.434	+0.6220	0.119	1	0.5	+0.791	0.228
00	*0.160*	*+1.074*		00	0.292	−0.5903	0.285	00	0.226	−1.317	0.145
01	*0.274*	*−0.358*		*01*	*0.274*	*−0.358*		*01*	*0.274*	*−0.358*	
10	*0.274*	*+0.358*		*10*	*0.274*	*+0.358*		*10*	*0.274*	*+0.358*	
11	0.292	−0.590	0.285	*11*	*0.160*	*+1.074*		11	0.226	+1.317	0.145
110	0.131	0.000	2.294	*000*	*0.160*	*−1.074*		000	0.066	−1.911	0.072
111	*0.160*	*−1.074*		001	0.131	0.000	2.294	*001*	*0.160*	*−1.074*	
								110	*0.160*	*+1.074*	
								111	0.066	+1.911	0.072
1100	*0.055*	*+1.791*		0010	0.077	−1.281	0.654	*0000*	*0.011*	*−2.511*	
1101	0.077	−1.281	0.654	*0011*	*0.055*	*+1.791*		*0001*	*0.055*	*−1.791*	
								1110	*0.055*	*+1.791*	
								1111	*0.011*	*+2.511*	
11010	0.022	0.000	6.306	*00100*	*0.055*	*−1.791*					
11011	*0.055*	*−1.791*		00101	0.022	0.000	6.306				
110100	*0.011*	*−2.511*		*001010*	*0.011*	*−2.511*					
110101	*0.011*	*+2.511*		*001011*	*0.011*	*+2.511*					

TAB. 6.1 – Definition of Huffman, *p-lex* Huffman and Hu-Tucker codes for a quantized Gaussian source. Leaves (e.g., alphabet symbols) are in italic.

and

$$\boldsymbol{\mu} = \{0.01091, 0.05473, 0.16025, 0.27411, 0.27411, 0.16025, 0.05473, 0.01091\}$$

respectively. The Huffman algorithm leads to the construction of the code

$$\{110100, 11011, 111, 01, 10, 00, 1100, 110101\}$$

detailed in table 6.1. The p-lex *algorithm proceeds as follows :*

Bit transition level	nodes	selected aggregation	expectation	probability
6	$\{a_1, a_8\}$	$(a_1, a_8) \rightarrow n_7$	$\mathbb{E}(n_7) = 0.000$	$P(n_7) = 0.022$
5	$\{n_7, a_2\}$	$(a_2, n_7) \rightarrow n_6$	$\mathbb{E}(n_6) = -1.281$	$P(n_6) = 0.077$
4	$\{n_6, a_7\}$	$(n_6, a_7) \rightarrow n_5$	$\mathbb{E}(n_5) = 0.000$	$P(n_5) = 0.131$
3	$\{n_5, a_3\}$	$(a_3, n_5) \rightarrow n_4$	$\mathbb{E}(n_4) = -0.590$	$P(n_4) = 0.292$
2	$\{n_4, a_4, a_5, a_6\}$	$(n_4, a_4) \rightarrow n_3$	$\mathbb{E}(n_3) = -0.478$	$P(n_3) = 0.566$
2	$\{a_5, a_6\}$	$(a_5, a_6) \rightarrow n_2$	$\mathbb{E}(n_2) = +0.622$	$P(n_2) = 0.434$
1	$\{n_2, n_3\}$	$(n_2, n_3) \rightarrow n_1$	$\mathbb{E}(n_1) = 0.000$	$P(n_1) = 1.000$

The reconstruction values obtained with this code are given in table 6.1. Note that both the Huffman code and the code constructed with the *p-lex* algorithm have an EDL of 2.521, while the source entropy is 2.471. The corresponding bit transition energies δ_j are also depicted. The reconstruction of symbols using the first bit only is improved by 1.57 dB (MSE is equal to 0.631 for the *p*-lex code instead of 0.906 for the Huffman code).

6.5.2 Hu-Tucker codes

For a given source, it may occur that the previous procedure leads to a code that preserves the lexicographic order in the binary domain. For example, it is well known that if the probability distribution function is a monotone function of symbol values, then it is possible to find a lexicographic code with the same compression efficiency as Huffman codes. But in general, it is not possible. In this section, Hu-Tucker [HT71] codes are used to enforce the lexicographic order to be preserved in the bit domain. The resulting codes may be sub-optimal, with the EDL falling into the interval $(h, h + 2($, where h denotes the entropy of the source. Thus, for the source $\mathbf{S}_{(2)}$, the EDL of the corresponding Hu-Tucker code is 2.583, which corresponds to a penalty in terms of EDL of 0.112 bit per symbol, against 0.050 for the Huffman code. The counterpart is that these codes have interesting progressivity features : the energy is concentrated on the first bit transitions (see Table 6.1). Thus, for the source $\mathbf{S}_{(2)}$, the reconstruction with the first bit only offers an improvement of 4.76 dB over Huffman codes.

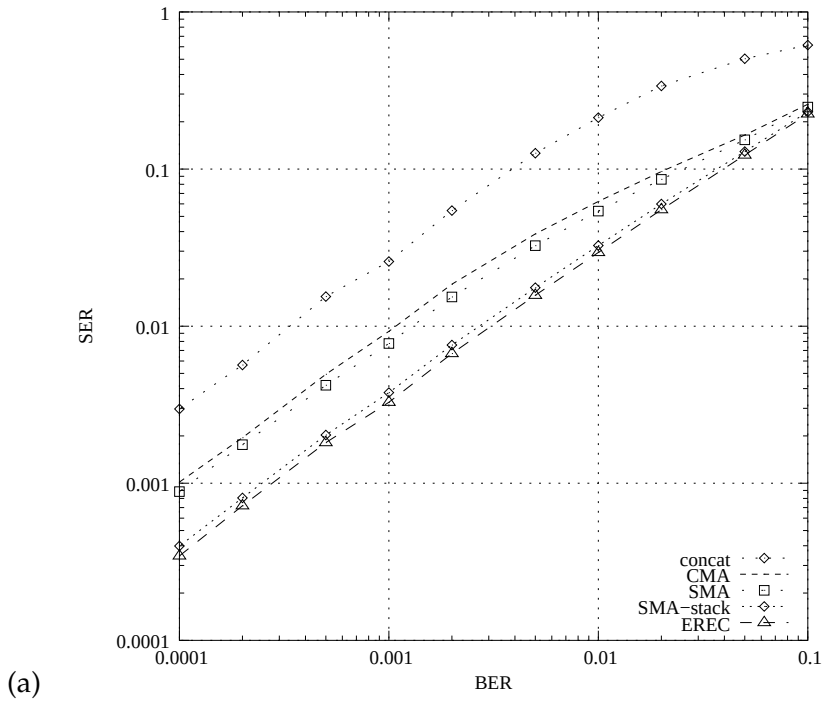
6.6 Simulation results

The performance of the different codes and BC algorithms has been assessed in terms of SER, SNR and Levenshtein distance with Source $\mathbf{S}_{(1)}$ and Source $\mathbf{S}_{(2)}$ (quantized Gaussian source), introduced in Ex. 6.1 and Ex. 6.5 respectively. Let us recall that the Levenshtein distance [Lev66] between two sequences is the minimum number of operations (e.g., symbol modifications, insertions and deletions) required to transform one sequence into the other. Unless the number of simulations is explicitly specified, the results shown are averaged over 100000 channel realizations and over sequences of 100 symbols. Since the source realizations are distinct, the number of emitted bits K_E is variable. Most of the algorithms that have been used to produce these simulation results are available on the web site [Jég].

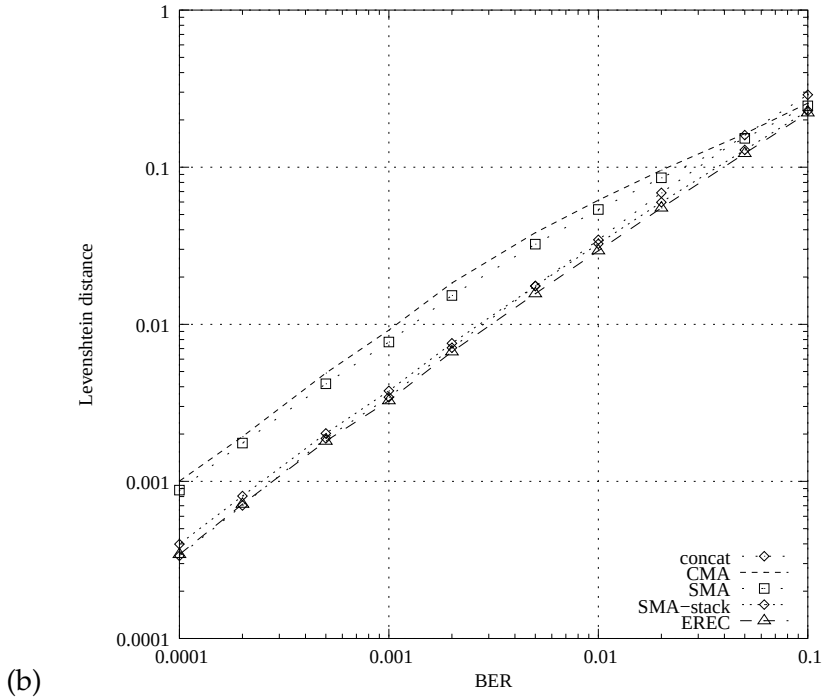
6.6.1 Error-resilience of algorithms CMA, SMA and SMA-stack for *p*-lex Huffman and Hu-Tucker codes

Fig. 6.5 and Fig. 6.6 respectively show for Source $\mathbf{S}_{(1)}$ and Source $\mathbf{S}_{(2)}$ the SER and the Levenshtein distance obtained with the algorithms CMA, SMA and SMA-stack in comparison with the concatenated scheme and a solution based on EREC [RK96] applied on a symbol (or codeword) basis, for channel error rates going from 10^{-4} to 10^{-1}. In Fig. 6.5, the results in terms of SER and normalized distance have been obtained with code $\mathcal{C}_5$ (cf. Ex. 6.1). Fig. 6.6 depicts the results obtained for a Huffman code optimized using the codetree optimization described in Section 6.5.1. This code is given in table 6.1.

In both figures, it appears that the concatenated scheme can be advantageously

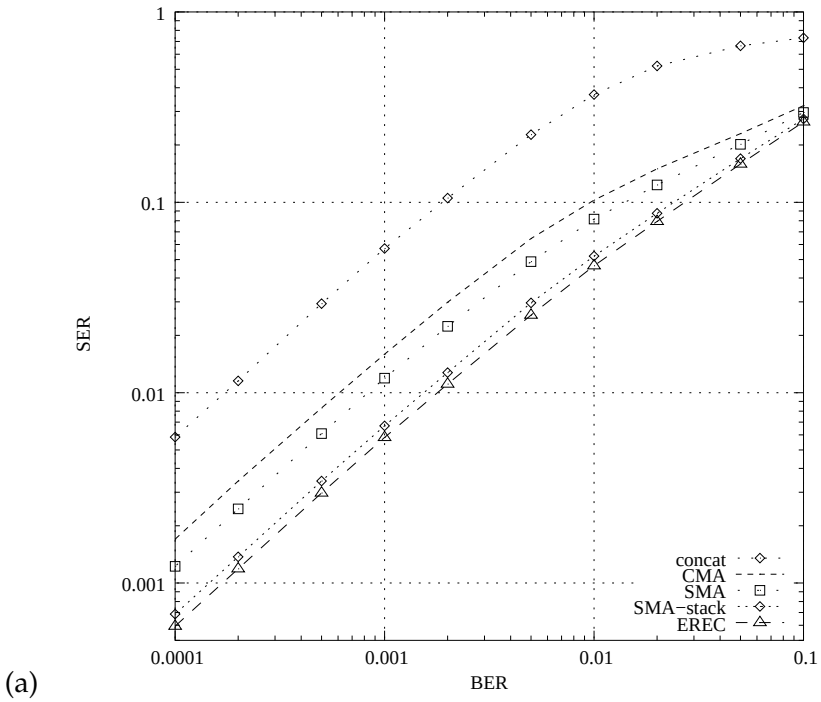


(a)

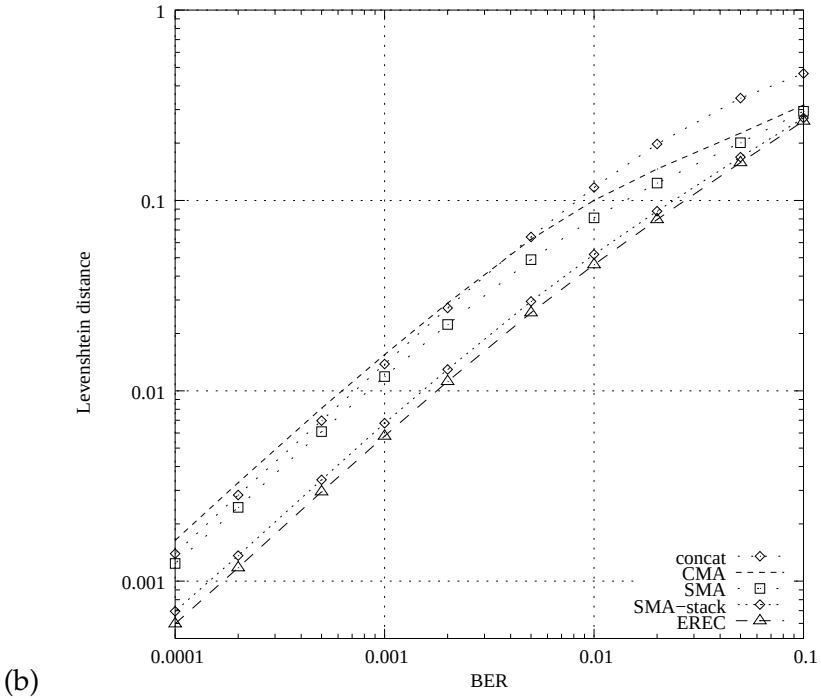


(b)

FIG. 6.5 – SER (a) and Levenshtein distance (b) performance of the different BC schemes for source $\mathbf{S}_{(1)}$ of Ex. 6.1 (code $\mathcal{C}_5$)



(a)



(b)

FIG. 6.6 – SER (a) and Levenshtein distance (b) performance of the different BC schemes with a quantized Gaussian source (source $\mathbf{S}_{(2)}$ encoded with Huffman codes).

replaced by the different BC algorithms described above. In particular, the SER performance of the SMA-stack algorithm approaches the one obtained with the EREC algorithm applied on a symbol basis (which itself already outperforms EREC applied on blocks of symbols), for a quite lower computational cost, since the complexity of the SMA-stack algorithm is linear with the sequence length. Similarly, it can be noticed in Fig. 6.7 that the best SNR values are obtained with Hu-Tucker codes used jointly with the EREC algorithm. It can also be noticed that the SMA-stack algorithm leads to very similar error-resilience performance. The results confirm that error propagation affects a smaller amount of reconstruction energy.

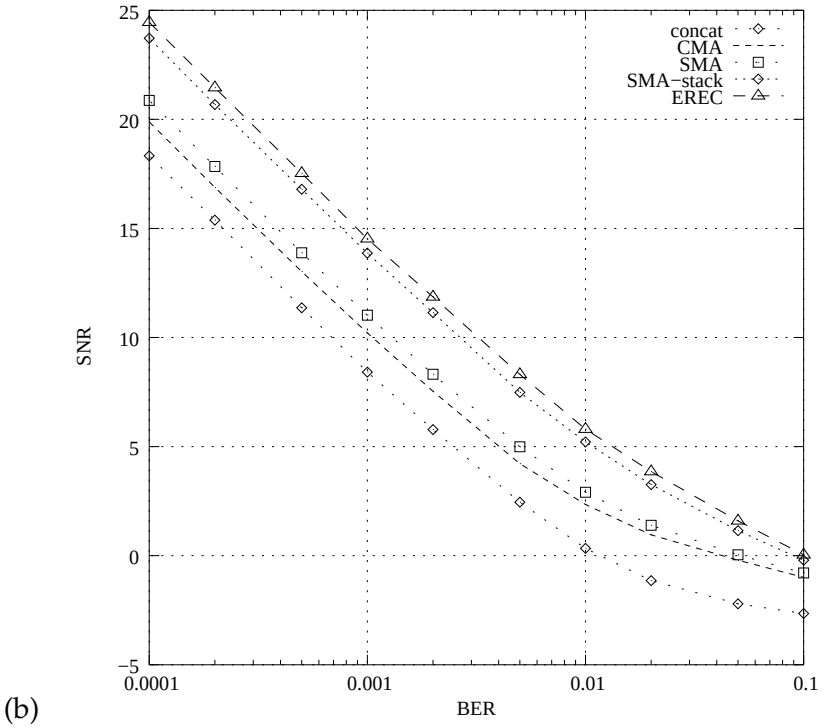
Remarks :

- for Source $\mathbf{S}_{(2)}$, Huffman and *p-lex* Huffman codes lead to the same error-resilience performance : the amount of energy conveyed by the bits mapped during the constant stage is identical for both codes. This can be observed in Table 6.1. However, for a large variety of sources, the *p-lex* Huffman codes lead to better results.
- The layered bitstream construction has not been included in this comparison : the layered bitstream construction offers improved error-resilience in a context of UEP. Simulation results depicted in Fig. 6.5, Fig. 6.6 and Fig. 6.7 assume that no channel code has been used.

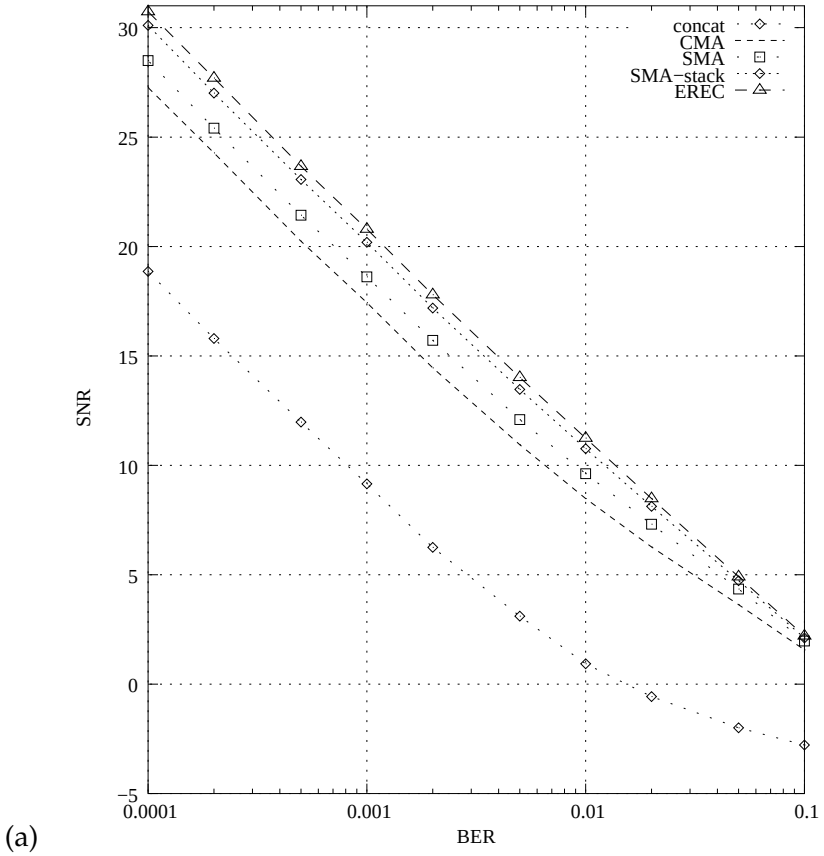
6.6.2 Progressivity performance of CMA and layered algorithms and impact of the code design

The amenability to progressive decoding (using expectation-based decoding) of the CMA and layered solutions with Huffman, *p*-lex and Hu-Tucker codes with respect to a concatenated scheme has also been assessed. These codes are also compared against lexicographic FLCs. for which the first bit transitions notably gather most of the energy. For the layered approach, nodes have been sorted according to the value of δ_j, under the constraint of Eqn. 6.7. Let us recall that the values of δ_j are provided in Table 6.1. The corresponding orders between nodes (identified by the path in the codetree) are given hereafter. The corresponding values of δ_j are also given.

Huffman	$\varnothing$	$\succ$	0	$\succ$	1	$\succ$	11	$\succ$	110	$\succ$	1101	$\succ$	11010
	0.022		0.477		0.224		0.285		2.294		0.654		6.306
p-lex Huffman	$\varnothing$	$\succ$	1	$\succ$	0	$\succ$	11	$\succ$	110	$\succ$	1101	$\succ$	11010
	0.297		0.119		0.013		0.285		2.294		0.654		6.306
Hu-Tucker	$\varnothing$	$\succ$	0	$\succ$	1	$\succ$	00	$\succ$	11	$\succ$	000	$\succ$	111
	0.626		0.228		0.228		0.145		0.145		0.072		0.072
FLC	$\varnothing$	$\succ$	0	$\succ$	1	$\succ$	01	$\succ$	10	$\succ$	00	$\succ$	11
	0.626		0.190		0.190		0.119		0.119		0.072		0.072



(b)



(a)

FIG. 6.7 – SNR performance of the different BC schemes (with pseudo-lexicographic Huffman) for *p*-lex Huffman codes (a) and Hu-Tucker codes (b).

The choice of the order has a major impact on the progressivity. Fig. 6.8 shows the respective MSE in terms of the number of bits received for different approaches. The performance obtained is better than the one obtained with a bit-plane FLC transmission, with, in addition, higher compression efficiency. The following points, identified on Fig. 6.8 with a number, are of interests :

1. The concatenation, which does not differentiate the bits, leads to the MSE performance that linearly decreases as the number of received bits increases.
2. Several curves converge to this point. This is due to the fact that the number of bits received at this point corresponds to the fixed length portions of Huffman and *p*-lex Huffman codes, and that these codes have similar energy concentration properties on the next transitions on the codetree (see Table 6.1).
3. For this source distribution, using the *p-lex* code instead of the usual Huffman code means transferring some *energy* from layer 2 (bits 101 to 200) to layer 1 (bits 1 to 100)
4. For the Huffman and *p*-lex Huffman codes, the bit transitions corresponding to nodes with a high depth in the codetree bear a lot of reconstruction energy. As a result, the concatenation of codewords gives better performance than the layered transmission in general.

To conclude, this figure shows the efficiency of Hu-Tucker codes transmitted with a layered BC scheme with respect to classical VLCs and transmission schemes. Note however that the performance gap between *p*-lex Huffman codes and Hu-Tucker codes strongly depends on the source statistical distribution.

6.6.3 Error-resilience with CMA

The performance in terms of SER of the CMA algorithm with an MPM decoder has been assessed against the one obtained with hard decoding and MPM decoding of concatenated codewords. The MPM decoder proceeds as described in Section 6.4.1. For this set of experiment, we have considered a quantized Gauss-Markov source with a correlation factor $\rho = 0.5$, and sequences of 100 symbols. Fig. 6.9 shows the significant gain (SER divided by a factor close to 2) obtained with the CMA structure with respect to the concatenated bitstream structure for the same decoding complexity (the number of states in the trellis is strictly the same for both approaches).

6.6.4 Turbo-decoding performance of CMA

In a last set of experiments, we have compared the amenability to turbo-decoding (using expectation-based decoding) of the CMA solution with respect to a concatenated scheme, both using Huffman codes. The decoding algorithm used for this simulation has been described in section 6.4.1 and section 6.4.2. We have considered the

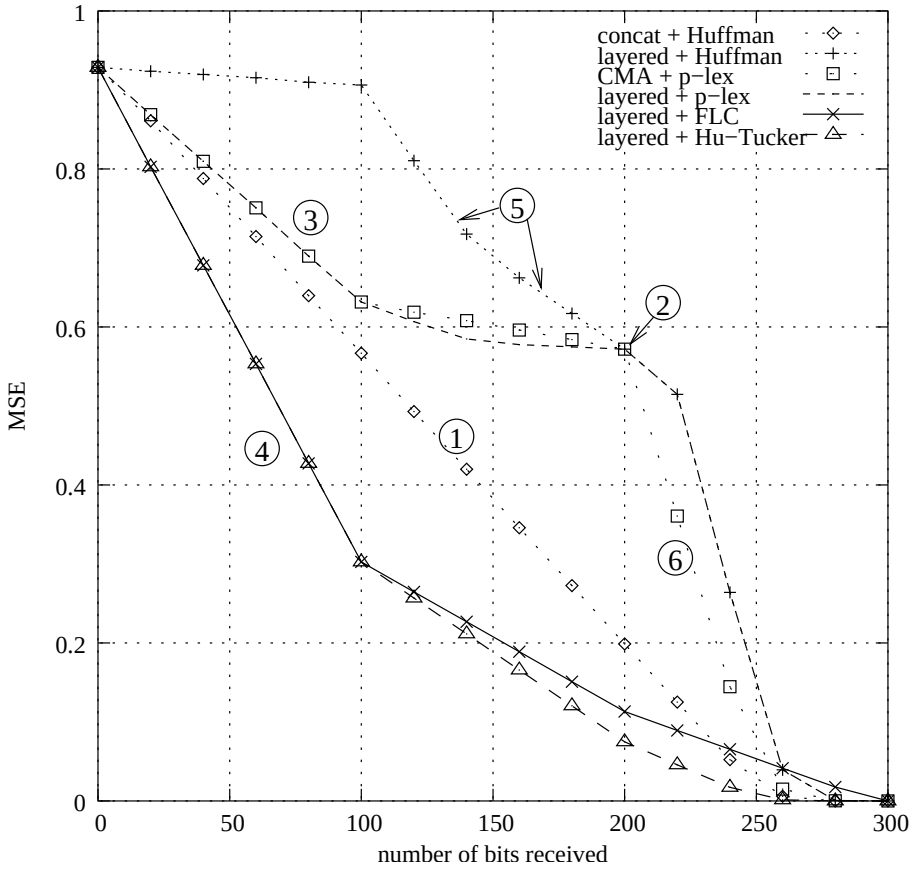


FIG. 6.8 – Progressive MSE performance/energy repartition profiles of the different codes and BC schemes without transmission noise.

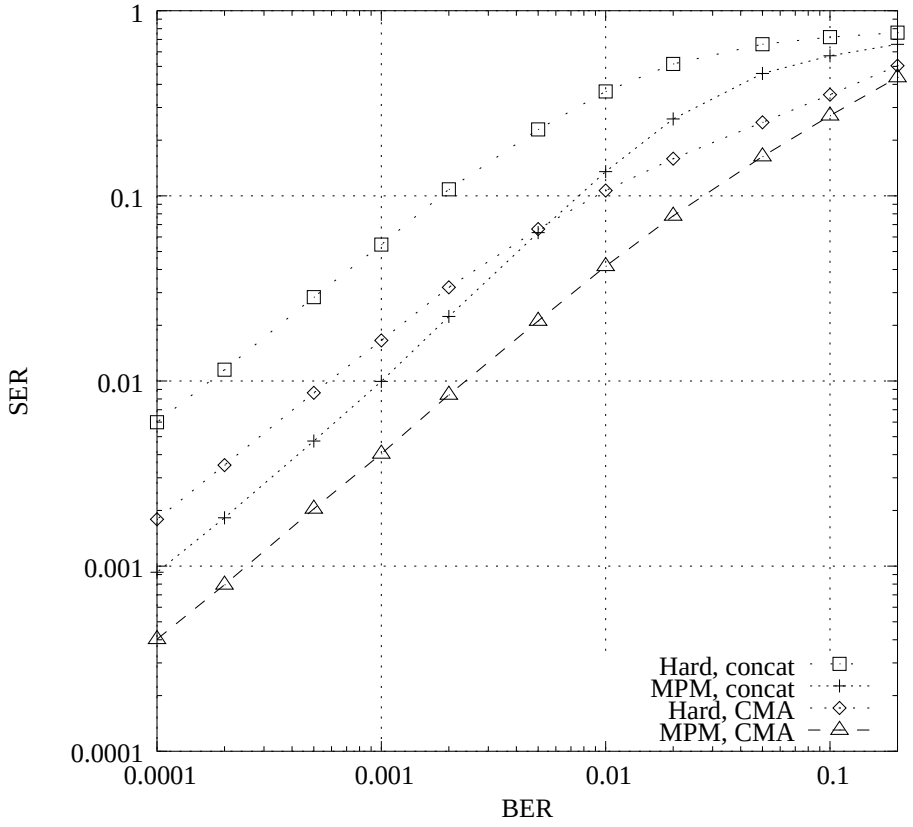


FIG. 6.9 – SER obtained by MPM decoding of a bitstream constructed by the CMA algorithm in comparison with those obtained with a concatenated bitstream structure (Gauss-Markov source with correlation $\rho = 0.5$).

first-order Markov source given in [MF98b]. This source takes its value in an alphabet composed of three symbols. The matrix of transition probabilities is defined as

$$\mathbb{P}(S_t/S_{t-1}) = \begin{bmatrix} 0.94 & 0.18 & 0.18 \\ 0.03 & 0.712 & 0.108 \\ 0.03 & 0.108 & 0.712 \end{bmatrix}.$$

The channel considered here is an AWGN channel. Results have been averaged over 10000 realizations of sequences of length 100. Fig. 6.10 shows the corresponding SER performance. This figure shows the clear advantage in terms of SER of the CMA algorithm. The improvement, expressed in terms of E_b/N_0, is around 0.4 dB. This gain is maintained as the number of iterations grows, and for the same complexity. For the Levenshtein distance measure, the concatenation leads to slightly better results. However, in multimedia applications, the SER is more critical than the Levenshtein distance.

6.7 Discussion and future work : error-resilient image coding and M-ary source binarization

In this section, a simple wavelet image coder is described and is used to depict the interest of an error-resilient entropy coder for error-resilient encoding of real sources. The benefits and limitations of the method for M-ary source binarization in state-of the art coders are also discussed.

6.7.1 Image coder

The bitstream construction algorithms as well as the code design have been experimented with images. We have considered a simple image coding system composed of a $9/7$ wavelet transform, a scalar uniform quantizer and an entropy coder based on VLCs. The image is first decomposed into 10 subbands using a three-stage wavelet transform. Four scalar quantizers have been used, depending on the wavelet decomposition level to which the quantizer applies. The low subband has been uniformly quantized on 192 cells. The number of quantization cells for the high subbands is respectively given by 97, 49, 25, from the lowest to the highest frequencies. The corresponding quantization steps are equal to 10.56, 10.45 and 10.24, respectively. The pdf of high frequency subbands is then modeled as a generalized Gaussian of the form

$$p(x) = \frac{1}{c(\alpha,\beta)} e^{-\frac{|x|}{\alpha}^{\beta}}, \tag{6.16}$$

where $\alpha > 0$ and $\beta > 0$ are the parameters of the generalized Gaussian and where $c(\alpha,\beta)$ is a normalization constant. Note that the pdf is Gaussian if $\beta = 2$. To estimate

BER	Huffman Concatenation	Huffman/Hu-Tucker CMA	Huffman/Hu-Tucker SMA-stack
10^{-4}	15.85	25.52	30.34
10^{-3}	11.16	22.25	25.76
10^{-2}	10.00	19.79	20.96

the parameters of the generalized Gaussian distribution, we use a gradient algorithm with the Kullback-Leibler distance d_K. This distance measures the overhead induced by the model. The values of α, β obtained for the high frequency subbands of the image Lena are depicted below.

subband	$LLLLHL$	$LLLLLH$	$LLLLHH$	$LLHL$	$LLLH$	$LLHH$	HL	LH	HH
α	8.335	5.350	4.310	4.220	3.546	3.524	3.173	3.661	3.621
β	0.540	0.562	0.527	0.585	0.630	0.658	0.735	0.951	1.133
d_K	0.0355	0.0222	0.0210	0.0136	0.0102	0.0036	0.0018	0.0003	0.0001.

The obtained Kullback-Leibler distance values confirm that the generalized Gaussian model is a first-order accurate model of the distribution of image subbands. The pdf obtained from the parameters can be synchronously computed on the encoder and on the decoder, assuming that the parameters are known on the decoder side. The probability laws are then used to design the VLC. For the low pass band, a Hu-Tucker code is chosen for the reasons exposed in section 6.5. For this subband, it appears that the Hu-Tucker code has the same EDL as the Huffman code. However, choosing this code for the high frequency dramatically increases the rate of the entropy coder, because the 0 symbol is assigned a codeword which is at least of length 2.

Hence, for the high frequencies, an Huffman code has been used. Our simple image coder gives a PSNR of 39.91 dB at a rate of 1.598 bit per symbol. These performance is obviously below the rate-distortion performance of state-of-the-art coders, but the comparative results in terms of error-resilience are however of interest. Note that, unlike UEP schemes, an error-resilient scheme such as the one proposed here does not require the knowledge of the channel characteristics. The gains achieved in terms of PSNR are given in Table 6.7.1. Each simulation point corresponds to the median value over 100 channel realizations for the image Lena. Fig. 6.11 also depicts significant improvements in terms of visual quality for typical realizations of the channel.

6.7.2 M-ary source binarization

The entropy coders used in most state-of-the-art image and video coding systems rely on a first binarization step followed by bit-plane (or bin) encoding using e.g. arithmetic coding. This general principle is currently applied in EBCOT [Tau99] and CA-

BC algorithm	linear complexity	mandatory parameters	soft decoding algorithms	main purpose
concatenation	yes	K or $L(\mathbf{X})$	bit/symbol trellis [BH00b], bit-level trellis [Bal97], etc	
CMA	yes	K	bit/symbol trellis	Error-resilience
SMA	yes	K and $L(\mathbf{X})$	?	Error-resilience
SMA-stack	yes	K and $L(\mathbf{X})$	?	error-resilience
layered	yes	K	?	progressivity
EREC	no	K and $L(\mathbf{X})$	?	Error-resilience

TAB. 6.2 – BC algorithms features.

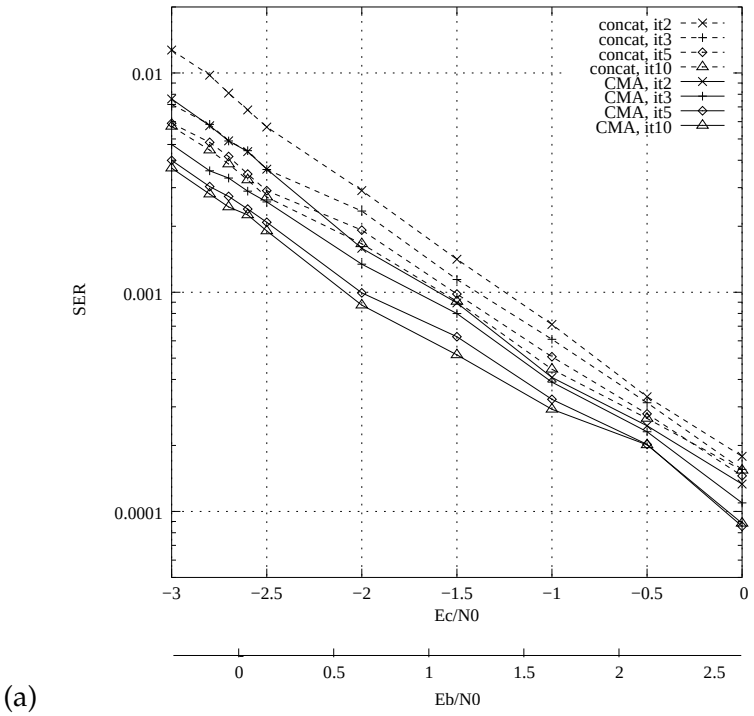
BAC [MSW03]. These algorithms will be all the most efficient both from a compression and error-resilience point of view if this binarization step makes use of a code which allows the concentration of most of the signal energy on the first transitions of the codetree of the binarization code. A priori analysis of the statistical distribution of the signal (transform coefficients, residue signals) to be encoded allows the design of a code with such properties. The scanning order of the bits resulting from the binarization step can then be defined as in section 6.3 above. This scanning order minimizing the dependencies between bit planes will contribute to improve the error-resilience of the entire coding approach.

If this binarization is coupled with an optimum entropy coder, the respective -entropy constrained- energy concentration properties of codes are modified. We noticed that in that particular case the FLCs and the Hu-Tucker Codes lead to similar rate-distortion curves. Moreover, the error sensitivity of the arithmetic coder to bit error suppresses the advantages of taking an error-resilient bitstream construction algorithm. Hence, the advantage of Hu-Tucker Codes in this context should only be considered from a complexity point of view. However, coupled with the bitstream construction algorithm, they can be an efficient alternative to the Exp-Colomb or CAVLC recommended in H.264 for low-power mobile devices [MSW03].

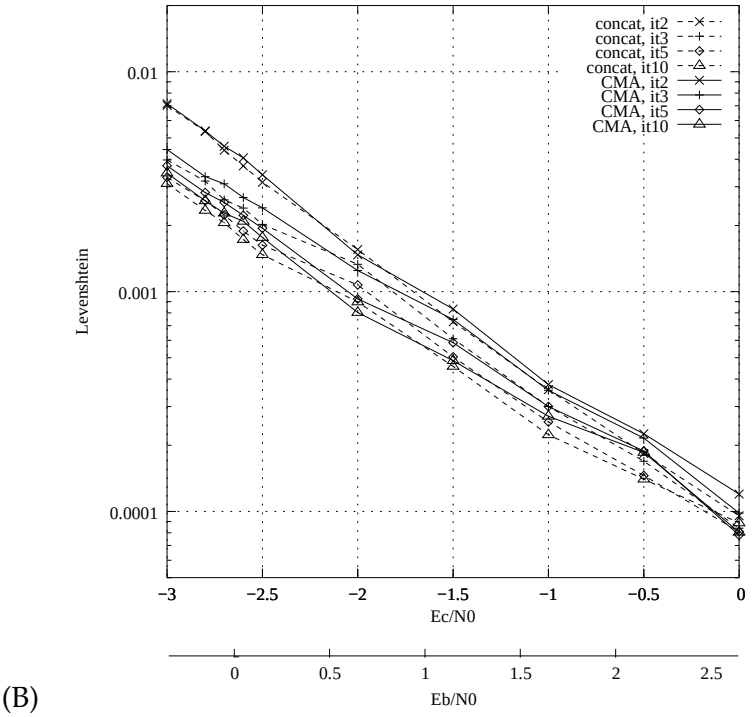
6.8 Conclusion

In this chapter, we have introduced a bitstream construction framework for the transmission of VLC-encoded sources over noisy channels. Several practical algorithms with different trade-offs in terms of error-resilience, progressivity and feasibility of soft decoding have been described. The features of these algorithms are summarized in Table. 6.2. Unlike the EREC algorithm, the complexity of the proposed algorithms increases linearly with the length of the sequence. For the CMA algorithm, the corresponding coding processes can be easily modeled under the form of stochastic automata which are then amenable for running MAP estimation and soft-decision

decoding techniques. Together with an MPM decoder, the CMA algorithm has been shown to increase the error-resiliency performance in terms of SER in comparison with a similar decoder for a concatenated bitstream, and this at no cost in terms of complexity. The approach has also been shown to offer improved SER performance in a turbo-VLC setup. For the other bitstream construction algorithms, i.e. SMA, EREC and SMA-stack, the design of efficient tractable soft and turbo decoding algorithms is a challenging problem. The code design has been shown to have a very high impact on the error-resilience, the progressivity and the amenability to enhance the UEP schemes. Hu-Tucker codes have been shown to be a good choice for these purposes. This framework can be applied to the problem of optimizing the source binarization step and in defining the appropriate scanning order of the bits resulting from the corresponding binarization step in modern image and video coders.



(a)



(B)

FIG. 6.10 – SER (a) and Levenshtein distance (b) obtained by two turbo-decoding schemes, both of them composed of a source coder and a RSCC : 1) a classical joint-source channel turbo-decoding, where the VLC codewords are concatenated ; 2) a joint-source channel where the VLC codewords are encoded/decoded with the CMA bitstream construction algorithm.

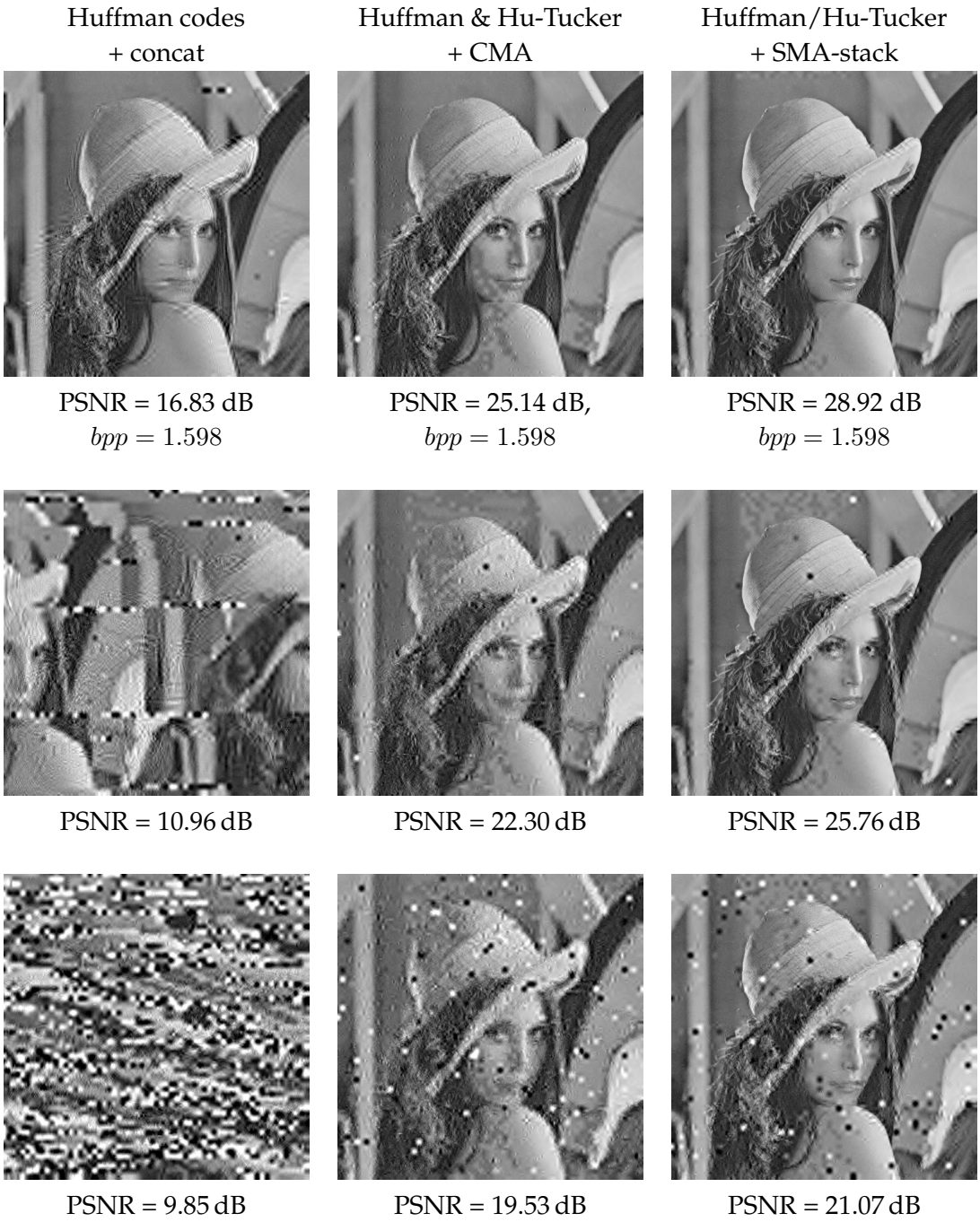


FIG. 6.11 – PSNR performance and visual quality obtained respectively with concatenated Huffman codes and two transmission schemes using Hu-Tucker codes for low subband and Huffman codes for high subbands. These schemes use respectively the CMA algorithm and the SMA-stack algorithm. The channel bit error rates are 0.0001 (top images), 0.001 (middle images) and 0.01 (bottom images).

Chapitre 7

Entropy coding with variable length re-writing systems

This chapter corresponds to a paper that will soon be submitted. It has been presented partially in [JG05b] and [JG05a]. It is also related to the work of [JG04c].

Abstract : This chapter describes a set of source codes for entropy coding of memoryless and Markovian sources. These codes are defined by sets of production rules of the form $a\bar{l} \rightarrow \bar{b}$, where a is a source symbol and $\bar{l}, \bar{b}$ are sequences of bits. The coding process can be modelled as finite state machines. A first code design method allowing preservation of the lexicographic order in the binary coded representation is described. With the construction method introduced, and given that there is a constraint on the number of states, codes can be obtained with a better compression efficency than Hu-Tucker codes. A second construction method allows to obtain codes such that the marginal bit probability converges to 0.5 as the sequence length increases. This is achieved even if the probability distribution function is not known by the encoder.

7.1 Introduction

Grammars are powerful tools which are widely used in computer sciences. Many lossless compression algorithms can be formalized with grammars. Codes explicitly based on grammars have been considered as a mean for data compression [KY00]. These universal codes losslessly encode a sequence in two steps. A first step consists in finding the production rules. A second step applies these rules to the sequence to be encoded. These codes have mainly been compared with dictionary-based compression algorithms such as LZ77 [ZL77] or [ZL78], which also implicitly use the grammar formalism. All these codes have in common the fact that the set of production rules depends on the data to be encoded.

In this chapter, a set of codes based on specific production rules is introduced. In contrast with grammar codes, the set of production rules is fixed given that the

source properties are known. These codes are implemented using finite state machines[1], (FSMs). and may be implemented with elementary finite-state machines (FSMs). FSMs have been considered in several contexts, many of them addressed in [Gor94]. They have been shown to be useful in a context of quantization [Eri05][GN98]. In [HV92], the authors show that finite-precision arithmetic codes can be implemented using FSMs, hence avoiding using any arithmetic operation. Indeed, only table lookup is required by these FSMs.

The form of production rules defining the code is presented in Section 7.2. The sequence of bits generated by a given production rule may be re-written by a subsequent production rule. The corresponding encoding and decoding FSMs are said to be *sequential*, which means that state transitions are triggered by a single symbol-input. Since their implementation only make use of table lookups, the complexity of the encoding and decoding procedures is the same as the one of Huffman codes [Huf52], which are encompassed by the proposed codes.

A possible drawback of these codes is that they require backward encoding. However, since most applications deal with block encoding, the forward encoding property is not absolutely required. The decoding and encoding procedures are described in Section 7.3. In Section 7.4 the compression efficiency is analyzed. It is shown with an example that the proposed codes allow for better compression efficiency than Huffman codes applied on a symbol basis, for similar memory and complexity requirements.

Two code construction methods are then described. Both methods lead to codes with the same EDL as the code from which they are constructed, e.g. Huffman codes. The first method constructs a set of production rules preserving the lexicographic order of the original source alphabet in the binary coded representation. This property is obviously of interest for database applications, since it allows the processing of comparative queries directly in the binary coded representation, hence avoiding prior decoding of the compressed dictionary for the query itself. Note that the lexicographic VLC of minimal expected length is obtained with the Hu-Tucker algorithm [HT71]. For some sources, the Hu-Tucker codes may have the same compression efficiency as Huffman codes, but it is not the case in general. Similar to Huffman codes, the best lexicographic codes for sequences are the generalized Hu-Tucker codes, i.e. the Hu-Tucker codes applied on the alphabet of sequences of symbols ordered by the corresponding lexicographical order. The number of nodes of the corresponding codetrees is of the order of the number of sequences, and such a solution can not be selected in practical systems. The method proposed in Section 7.5 constructs lexicographic codes with at least the same compression performance as Huffman codes. We show how the resulting codes can outperform, for the compression efficiency/number of states com-

[1] In computer sciences, the term *transducer* is used instead of the term FSM. It is worth noting that Shannon used the term transducer in [Sha48]. In the coding community, FSMs are also referred to as to *automaton*, although this term is less precise.

promise, the Hu-Tucker codes and their generalized version applied on a block basis.

The second construction method described in Section 7.6 allows to obtain codes, for stationary sources, such that the marginal bit probability is equal to 0.5. The main advantage of these codes is that this probability is equal to 0.5 even if the actual source probabilities are not known at the encoder, or if the assumed *a priori* probabilities differ from the true probabilities. Since channel encoders widely assume that 0*s* and 1*s* have the same probability, this property is of interest when compressed bitstreams protected by such encoders are transmitted over noisy channels. Indeed, for a transmission scheme that makes use of a systematic error correcting code, a mismatch in the marginal bit probability leads to a capacity loss [ZA02]. In Table I of [ZABM04], the gap to the optimal performance theoretically achievable has been shown to be quite important on Gaussian and Rayleigh channels. This observation has motivated the authors to use nonsystematic turbo-codes for highly-biased sources [ZABM04]. In this paper, this problem is addressed from a source soft decoding perspective.

7.2 Notation and definitions

In the sequel random variables are denoted by upper case and the corresponding realizations are denoted by lower case. Sets are denoted by calligraphic characters. The cardinality of a given set $\mathbf{X}$ is denoted $|\mathbf{X}|$. The concatenation of two sequences x and y is denoted xy. The t^{th} element of a sequence x is denoted x_t. The void sequence is denoted ε. This sequence is the neutral element for the concatenation, i.e. $\forall x,\ \varepsilon x = x\varepsilon = x$. The set of sequences of elements of $\mathbf{X}$ with length i is denoted $\mathbf{X}^i$. The length of a given sequence x is denoted $L(x)$. A sequence x is said to be a prefix of a sequence y if and only if $L(x) \leq L(y) \wedge \forall i \in \{1 \dots L(x)\},\ x_i = y_i$. In the following we write $x \sqsubset y$ if x is a prefix of y, $x \not\sqsubset y$ otherwise. We define $\mathbf{X}^+ = \bigcup_{i=1}^{\infty} \mathbf{X}^i$ and $\mathbf{X}^* = \{\varepsilon\} \cup \mathbf{X}^+$. Hence $\mathbf{X}^*$ denotes the set of sequences composed of elements of $\mathbf{X}$.

Let $\mathbf{S} \in \mathcal{A}^+$ be a sequence of source symbols taking their values in a finite alphabet $\mathcal{A} = \{a_1, \dots, a_i, \dots\}$. The alphabet $\mathcal{A}$ is assumed to be ordered according to a total order $\prec$. Without loss of generality, we assume that $a_1 \prec a_2 \prec \dots \prec a_i \prec \dots \prec a_{|\mathcal{A}|}$. Let us define $\mathcal{B} = \{0, 1\}$. In the sequel, the emitted bitstream is denoted $\mathbf{E} = E_1 \dots E_{L(\mathbf{E})} \in \mathcal{B}^*$ and its realization is denoted $\mathbf{e} = e_1 \dots e_{L(\mathbf{e})}$.

Définition 7.1 *A variable length re-writing system (VLRS) is a set $\mathcal{R} = \bigcup_{i \in \mathcal{A}} \mathcal{R}_i$, where $\mathcal{R}_i$ denotes the set of rules related to a given symbol a_i, defined as*

$$
\begin{array}{lllll}
r_{1,1}: & a_1\,\bar{l}_{1,1} & \rightarrow & \bar{b}_{1,1} = b_{1,1}^1 \dots b_{1,1}^{L(\bar{b}_{1,1})}, \\
 & & \vdots & & \\
r_{i,j}: & a_i\,\bar{l}_{i,j} & \rightarrow & \bar{b}_{i,j} = b_{i,j}^1 \dots b_{i,j}^{L(\bar{b}_{i,j})}, \\
 & & \vdots & & \\
r_{|\mathcal{A}|,|\mathcal{R}_{|\mathcal{A}|}|}: & a_{|\mathcal{A}|}\,\bar{l}_{|\mathcal{A}|,|\mathcal{R}_{|\mathcal{A}|}|} & \rightarrow & \bar{b}_{|\mathcal{A}|,|\mathcal{R}_{|\mathcal{A}|}|}
\end{array}
$$

where $\bar{l}_{i,j} \in \mathcal{B}^*,\ \bar{b}_{i,j} \in \mathcal{B}^+$. *This set is such that*

1. *The set* $\bigcup_{i=1}^{|\mathcal{A}|}\bigcup_{j=1}^{|\mathcal{R}_i|}\{\bar{b}_{i,j}\}$ *forms a prefix code (i.e. no codeword is the prefix of another codeword, see Chapter 5 of [CT91]) ;*

2. $\forall j,\ \bigcup_{i=1}^{|\mathcal{R}_i|}\{\bar{l}_{i,j}\}$ *is the set* $\{\varepsilon\}$ *or forms a full prefix code (i.e, such that the Kraft sum is equal to* 1*) ;*

3. $\forall i, i'/i' \neq i, \forall j, j',\ \bar{b}_{i,j} = \bar{l}_{i',j'}$ *or* $\bar{b}_{i,j} \not\sqsubset \bar{l}_{i',j'}$.

These production rules allow successive transformation of a sequence $\mathbf{s}$ of symbols into a sequence $\mathbf{e}$ of bits. These rules are assumed to be reversible : inverting the direction of the arrow allows to recover a given sequence $\mathbf{s}$ from the corresponding bitstream $\mathbf{e}$. Note that a given production rule absorbs a symbol (a_i) and some bits ($\bar{l}_{i,j}$) and generates a given sequence of bits ($\bar{b}_{i,j}$).

The restricting assumptions proposed in Definition 7.1 are motivated as follows. Condition 1 ensures that the code is uniquely decodable without any ambiguity in the forward direction. Condition 2 ensures that, for any possible realization $\overline{x}$ of the sequence of bits following the symbol $S_t = a_i$ to be encoded, a production rule of the form $a_i\overline{y} \rightarrow \bar{b}$ such that $\overline{y} \sqsubset \overline{x}$ can be chosen to encode the realization of S_t. The encoding initialization step also ensures that the number of bits following the last symbol is sufficient to trigger at least one rule.

Note that from Condition 2 we deduce that $\forall i,\ |\mathcal{R}_i| \geq 1$. Hence, at least one production rule will be assigned to each symbol. Note that if $|\mathcal{R}_i| = 1$ we have $\bar{l}_{i,1} = \varepsilon$. More generally, a VLRS is a Fixed-to-Variable (F-to-V) Length code if $\forall i\ |\mathcal{R}_i| = 1$ and $\bar{l}_i = \{\varepsilon\}$. Condition 3 is introduced to restrict the analysis to VLRS verifying the following condition : the knowledge of the rule applied for symbol s_t is sufficient to select the rule for the encoding of symbol s_{t-1}. This condition, which is not strictly required to define a valid encoding/decoding system, is used by the proof of Property 7.6 in Section 7.4.

Note that the usual Fixed-to-Variable length codes such as Huffman or Shannon codes, are covered by Definition 7.1. They correspond to the subset of VLRSs such that $\forall i\ |\mathcal{R}_i| = 1$.

Example 7.1: The code $\mathcal{C}_1 = \{0, 10, 11\}$ can be regarded as the VLRS

$$\begin{array}{lrcl} r_{1,1}: & a_1 & \rightarrow & 0 \\ r_{2,1}: & a_2 & \rightarrow & 10 \\ r_{3,1}: & a_3 & \rightarrow & 11. \end{array}$$

Definition 1 does not warranty that the system defines a valid prefix code. For example, a rule $r_{i,j}$ where $\overline{b}_{i,j} \sqsubset \overline{l}_{i,j}$ is not valid. In this chapter, we will only consider VLRS defining valid prefix codes. For this purpose, it will also be assumed that the encoding process is ergodic. We will come back on this point in section 7.4. The suffix-constrained codes introduced in [JG04c] form a subset of VLRS and are now characterized as follows.

Définition 7.2 *A suffix-constrained code is a VLRS such that $\forall i, j$, $\overline{l}_{i,j}$ is a suffix of $\overline{b}_{i,j}$.*

By construction, a production rule of a suffix-constrained code generates the same bits as those it has absorbed. Hence, the following property :

Property 7.1: A bit generated by a production rule of a suffix-constrained code will not be modified by a subsequent production rule.

Example 7.2: The following VLRS $\mathcal{C}_2$ is a suffix-constrained code :

$$\begin{array}{lrcl} r_{1,1}: & a_1 0 & \rightarrow & 10 \\ r_{1,2}: & a_1 1 & \rightarrow & 01 \\ r_{2,1}: & a_2 & \rightarrow & 00 \\ r_{3,1}: & a_3 & \rightarrow & 11. \end{array}$$

Note that Code $\mathcal{C}_2$ can not be encoded in the forward direction[2]. We will come back on this point in Section 7.3. The lexicographic code $\mathcal{C}_3$ defined as

$$\mathcal{C}_3 = \left\{ \begin{array}{lrcl} r_{1,1}: & a_1 & \rightarrow & 00 \\ r_{2,1}: & a_2 0 & \rightarrow & 01 \\ r_{2,2}: & a_2 1 & \rightarrow & 10 \\ r_{3,1}: & a_3 & \rightarrow & 11 \end{array} \right.$$

and the code $\mathcal{C}_4$ defined as

$$\mathcal{C}_4 = \left\{ \begin{array}{lrcl} r_{1,1}: & a_1 1 & \rightarrow & 0 \\ r_{1,2}: & a_1 0 & \rightarrow & 10 \\ r_{2,1}: & a_2 & \rightarrow & 110 \\ r_{3,1}: & a_3 & \rightarrow & 111 \end{array} \right.$$

[2]Theoretically, it can. However, the dimension of the encoding FSM with respect to the number of states is not tractable.

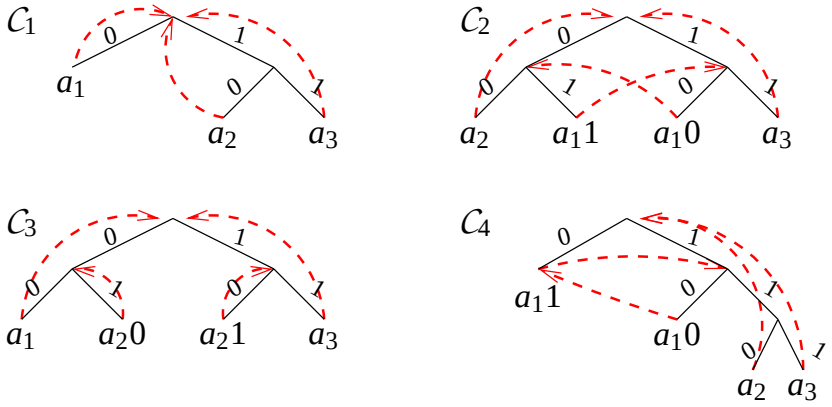


FIG. 7.1 – Tree and FSM representation of $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$. The transitions triggered by the production rules are depicted by arrows.

will also be considered in the sequel. This code $\mathcal{C}_4$ allows encoding the symbol a_1 with less than 1 bit. Note that these codes are not suffix-constrained codes.

VLRS can also be represented using trees, as depicted in Fig. 7.1. The tree structure corresponds to the one of the prefix code defined by $\bigcup_{i=1}^{|\mathcal{A}|}\bigcup_{j=1}^{|\mathcal{R}_i|}\{\bar{b}_{i,j}\}$. Leaves correspond to both the symbols $\{a_i\}_i$ and the sequences of bits $\{\bar{l}_{i,j}\}_{i,j}$.

7.3 Encoding and decoding FSMs

On the encoder side, the production rules transform the sequence $\mathbf{s}$ into the sequence $\mathbf{e}$ of bits. Any segment of the sequence (composed of symbols *and* bits) can be rewritten if there exists a rule having this segment as an input (this input is composed of one symbol *and* a variable number of bits). In general, the set of rules defining a VLRS does not allow encoding the sequence $\mathbf{S}$ in the forward direction with a sequential FSM of reasonable dimension. Therefore, the encoding must be processed backward. To initiate the encoding process, a specific rule is used to encode the last symbol of the sequence. Indeed the last symbol may not be sufficient to trigger a production rule by itself. For this reason, some termination bits are concatenated to the sequence of symbols. The sequence of termination bits and its realization are denoted $\mathbf{U}$ and $\mathbf{u}$ respectively. They can be arbitrarily defined at the condition that the termination bit(s) do(es) not trigger a production rule by itself. Hence, the choice $\mathbf{u} = 0$ is valid for the codes $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ but should not be used for code $\mathcal{C}_4$, since 0 triggers the rule $r_{1,1}$.

Property 7.2: Transmitting the termination bit(s) $\mathbf{U}$ is not required for suffix-constrained codes.

Proof: Let us assume without loss of generality that the termination constraint $\mathbf{U}$ is arbitrarily chosen to be a sequence of 0. From Property 7.1, we deduce that $\mathbf{U}$ is the suffix of the intermediate re-written terms. Therefore $\mathbf{U}$ is also the suffix of the emitted bitstream $\mathbf{E}$. Hence, these bits are deterministically known on the decoder side and need not be transmitted. ■

Example 7.3: Let $\mathbf{s}_1 = a_1a_2a_2a_3a_2a_1a_1a_1$ be a sequence of symbols taking their values in the alphabet $\mathcal{A}_1 = \{a_1, a_2, a_3\}$. This sequence is encoded with the code $\mathcal{C}_2$. Since the last symbol is a_1, no rule applies directly. Therefore, the termination bit $\mathbf{u} = 0$ is concatenated to this sequence in order to initiate the encoding. The encoding then proceeds as follows :

$$\begin{aligned}
r_{1,1}: \quad & \mathbf{s}_1\mathbf{u} = a_1a_2a_2a_3a_2a_1a_1\underline{a_1}0 \\
r_{1,2}: \quad & a_1a_2a_2a_3a_2a_1\underline{a_1}10 \\
r_{1,1}: \quad & a_1a_2a_2a_3a_2\underline{a_1}010 \\
r_{2,1}: \quad & a_1a_2a_2a_3\underline{a_2}1010 \\
r_{3,1}: \quad & a_1a_2a_2\underline{a_3}001010 \\
r_{2,1}: \quad & a_1a_2\underline{a_2}11001010 \\
r_{2,1}: \quad & a_1\underline{a_2}0011001010 \\
r_{1,1}: \quad & \underline{a_1}00011001010 \\
& \mathbf{e}_1\mathbf{u} = 1000011001010
\end{aligned}$$

Example 7.3 illustrates Property 7.2. The termination bit $\mathbf{u} = 0$ is not modified by the successive applications of the production rules. This bit is deterministically known on the decoder side. Hence, the suffix does not need to be transmitted. Since these termination bits may be required in the general case of VLRS, it will be assumed in the following that they are known at the decoder. In the following example, the termination bit must be $\mathbf{u} = 1$.

Example 7.4: Let us now consider the sequence $\mathbf{s}'_1 = a_1a_1a_1a_1a_1$. This sequence is encoded with code $\mathcal{C}_4$ as

$$\begin{aligned}
r_{1,1}: \quad & \mathbf{s}'_1\mathbf{u}' = a_1a_1a_1a_1\underline{a_1}1 \\
r_{1,2}: \quad & a_1a_1a_1\underline{a_1}0 \\
r_{1,1}: \quad & a_1a_1\underline{a_1}10 \\
r_{1,2}: \quad & a_1\underline{a_1}00 \\
r_{1,1}: \quad & \underline{a_1}100 \\
& \mathbf{e}'_1 = 000
\end{aligned}$$

Note that the sequence is encoded with less than 1 bit per symbol. The decoding is processed forward using the reverse rules. The encoding and decoding algorithms are

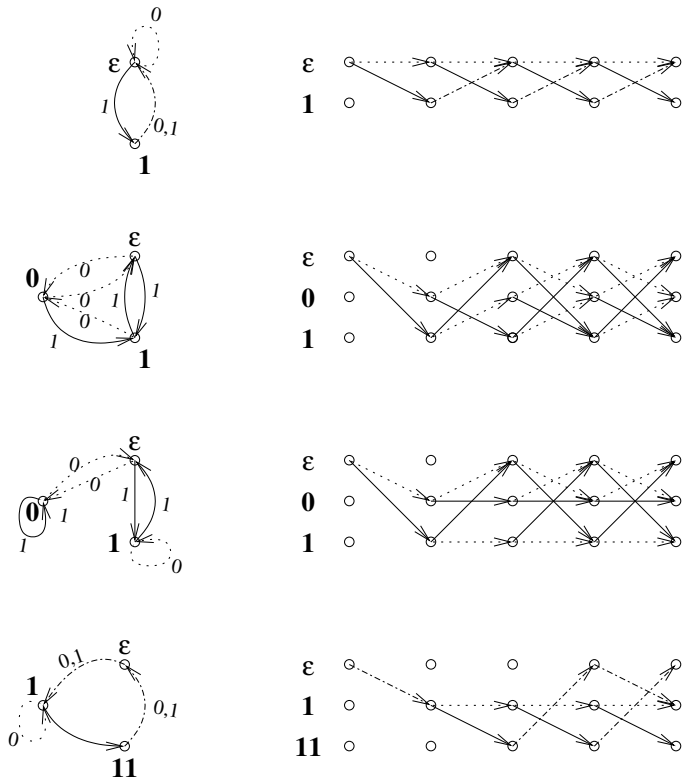


FIG. 7.2 – Decoding FSMs corresponding to the codes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$ and corresponding decoding trellises. Transitions corresponding to 0*s* and 1*s* are represented with dotted and solid lines respectively.

implemented using FSMs. These FSMs are used to catch the memory of the encoding and decoding processes. They have to be constructed so that the states include the knowledge of the bits that may be re-written or used to select the next production rule. The transitions on the FSM representing the encoding process are triggered by symbols. The internal states of the encoding FSM are given by the variable length segments of bits $\{\bar{l}_{i,j}\}$. This FSM may be simplified if a variable length bit segment $\bar{l}_{i,j}$ is a prefix of another segment $\bar{l}_{i',j'}$ (in that case, according to Definition 7.1, we have $i \neq i'$). For the code $\mathcal{C}_1$, we have $\forall i, j,\ \bar{l}_{i,j} = \varepsilon$. Therefore, there is only one internal state $\{\varepsilon\}$ for the encoding FSM corresponding to the code $\mathcal{C}_1$. The sets of states of encoding FSMs of codes $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$ are identical and are equal to $\{\mathbf{0}, \mathbf{1}\}$. Note that the termination sequence $\mathbf{u}$ allows to uniquely identify the starting state of the encoding FSM.

The states of the decoding FSMs correspond to bit segments that have already been decoded, but which are not sufficient to identify a symbol. For VLCs such as Huffman codes, these internal states correspond to the internal nodes of the codetree.

Example 7.5: The set of internal states of codes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$ are respectively $\{\varepsilon, \mathbf{1}\}$, $\{\varepsilon, \mathbf{0}, \mathbf{1}\}$, $\{\varepsilon, \mathbf{0}, \mathbf{1}\}$ and $\{\varepsilon, \mathbf{1}, \mathbf{11}\}$.

Since the bit segments that are not sufficient to identify a symbol correspond to the set of strict[3] prefixes of the set of codewords $\bigcup_{i,j} \bar{b}_{i,j}$, we deduce the following property.

Property 7.3: If the prefix code $\bigcup_{i,j} \bar{b}_{i,j}$ is a full prefix code, then the number of states of the decoding FSM is equal to $|\mathcal{R}| - 1$.

Proof: Since the number of internal nodes of a full prefix code is also equal to the number of strict prefixes of the code, the property straigthforwardly stems from the classical property that the number of internal nodes of a full prefix code composed of n symbols is equal to $n - 1$. ■

According to Property 7.3, the number of production rules is directly linked to the number of states. Consequently, it is of interest to keep the number of rules as low as possible. For this purpose, the following property may be used to reduce the number of production rules without modifying the function defined by the FSM.

Property 7.4: Let $r_{i,j}$ and $r_{i,j'}$ be two production rules such that

$$\begin{cases} l_{i,j}^1 \dots l_{i,j}^{L(\bar{l}_{i,j})-1} &= l_{i,j'}^1 \dots l_{i,j'}^{L(\bar{l}_{i,j'})-1} \\ b_{i,j}^1 \dots b_{i,j}^{L(\bar{b}_{i,j})-1} &= b_{i,j'}^1 \dots b_{i,j'}^{L(\bar{b}_{i,j'})-1} \\ l_{i,j}^{L(\bar{l}_{i,j})} &= b_{i,j}^{L(\bar{b}_{i,j})}, \end{cases}$$

then these rules can be merged into a single rule

$$a_i \bar{l}_{i,j}^1 \dots \bar{l}_{i,j}^{L(\bar{l}_{i,j})-1} \rightarrow \bar{b}_{i,j'}^{L(\bar{b}_{i,j'})-1}$$

which is equivalent to the set of two rules.

The graphical representations of the decoding FSMs may be deduced from the tree representations given in Fig. 7.1. These FSMs and the corresponding trellises are depicted in Fig. 7.2. For sake of clarity, the symbols generated by the bit transitions are not shown. However, note that the set of generated symbol(s) must also be associated with each bit transition. For the codes $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$, at most 1 symbol is associated with each bit transition. It is not the case for the code $\mathcal{C}_4$, where the transition starting from decoding state **1** triggered by the bit 0 generates the symbol a_1 twice. As shown in Example 7.4 and demonstrated in Section 7.4, this transition allows to encode long sequences of a_1 with less than 1 bit, at the cost of a higher encoding cost for the symbols a_2 and a_3.

[3] A codeword x is said to be a strict prefix of y if $x \sqsubset y$ and $x \neq y$.

7.4 Compression efficiency

This section analyzes the compression efficiency of VLRSs for a memoryless stationary source. First, the asymptotic EDL is given when the sequence length increases to infinity. This quantity is denoted EDL_∞ in the following. Next, we discuss the compression efficiency for sequences of finite length.

Let us assume that $\mathbf{S}$ is an independent identically distributed (i.i.d.) source characterized by its stationary PMF on $\mathcal{A}$:

$$\boldsymbol{\mu} = \{\mathbb{P}(a_1), \dots, \mathbb{P}(a_i), \dots\}.$$

Let

$$\delta(r_{i,j}) \triangleq L(\bar{b}_{i,j}) - L(\bar{l}_{i,j}) \tag{7.1}$$

denote the number of bits generated by a given production rule $r_{i,j}$.

Property 7.5: Let $\mathcal{C}$ be a VLRS such that $\forall i,\ \forall j, j'\ \delta(r_{i,j}) = \delta(r_{i,j'})$. Then we have

$$\mathrm{EDL}_\infty(\mathcal{C}) = \sum_{a_i \in \mathcal{A}} \mathbb{P}(a_i)\delta_{i,1}. \tag{7.2}$$

Proof: Let $S_t = a_i$ be the symbol to be encoded at instant t. This symbol is encoded using a rule $R_t = r_{i,j}$. From the assumption $\forall j, j'\ \delta(r_{i,j}) = \delta(r_{i,j'})$, we deduce that the number of bits generated by the rule r_t does not depend on j, hence $\delta(r_{i,j}) = \delta(r_{i,1})$. Then we have $\mathbb{E}(\delta(R_t)) = \sum_{a_i \in \mathcal{A}} \mathbb{P}(a_i)\delta_{i,1}$. Note that this expectation does not depend on the instant t. Therefore, the expectation of the bitstream length $L(\mathbf{E})$ is given by $L(\mathbf{S}) \times \mathbb{E}(\delta(R_t)) + \mathbb{E}(L(\mathbf{U}))$. The expectation $\mathbb{E}(L(\mathbf{U}))$ of the length of the termination constraint is bounded by the quantity $\max_{i,j} l_{i,j}$, which does not depend on the length $L(\mathbf{S})$. Consequently,

$$\mathrm{EDL}_\infty(\mathcal{C}) = \lim_{L(\mathbf{S})\to\infty} \frac{1}{L(\mathbf{S})} \left(L(\mathbf{S}) \times \mathbb{E}(\delta(R_t)) + \mathbb{E}(L(\mathbf{U}))\right) \tag{7.3}$$

$$= \sum_{a_i \in \mathcal{A}} \mathbb{P}(a_i)\delta_{i,1}. \tag{7.4}$$

■

Note that the proof of Property 7.5 does not require that the source is memoryless.

Example 7.6: Let us assume that $\mathbf{S}$ *is a source of stationary probabilities* $\boldsymbol{\mu}_1 = \{0.7, 0.2, 0.1\}$. *The entropy of this source is* 1.157. *The expected length of the code* $\mathcal{C}_1$ *is equal to* 1.3. *For the code* $\mathcal{C}_2$, *we have* $\delta(r_{1,1}) = \delta(r_{1,2}) = 1$ *and* $\delta(r_{2,1}) = \delta(r_{3,1}) = 2$. *The expected length of this code is also equal to* 1.3.

Property 7.6: Let $\mathcal{C}$ be a VLRS and $\mathbf{S}$ an i.i.d. source. The process

$$(Z_{t'})_{t'} = (R_{L(\mathbf{S})}, \ldots, R_t, \ldots, R_1)$$

obtained from the process $(R_t)_t$ by reversing the symbol instant t as $t' = L(\mathbf{S}) - t + 1$ forms a homogeneous Markov chain of transition probabilities

$$\lambda_{i,j,i',j'} \triangleq \mathbb{P}(Z_{t'} = r_{i,j} | Z_{t'-1} = r_{i',j'}) \tag{7.5}$$

$$= \begin{cases} \mathbb{P}(a_i) & \text{if } \overline{l}_{i,j} \sqsubset \overline{b}_{i',j'}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.6}$$

Proof: Let $R_t : S_t\overline{L}_t \rightarrow \overline{B}_t$ denote the rule used to encode the symbol S_t. From condition 3 of definition 7.1 we deduce that the tuple $(\overline{B}_{t+1}, S_t)$ and *a fortiori* the tuple (R_{t+1}, S_t) suffice to identify the rule R_t to trigger. Therefore, knowing the realization of R_{t+1}, the probabilities of the variable R_t are governed by the statistics of the source $\mathbf{S}$. From the memoryless assumption on the source $\mathbf{S}$, we deduce that the probabilities of the event R_t are fully defined by the knowledge of the event R_{t+1}, which leads to conclude that Z_t is a Markov chain. With the property that the elements of $\mathbf{S}$ are identically distributed, we also deduce that this chain is homogeneous. Note that the event $R_t = r_{i,j}$ occurs if and only if $S_t = a_i \wedge \overline{l}_{i,j} \sqsubset \overline{b}_{i',j'}$, where the codeword $\overline{b}_{i',j'}$ is the realization of $\overline{B}_{t+1}$ and has been generated by the previous production rule (at instant $t+1$). As a consequence, the probability $\lambda_{i,j,i',j'}$ is deduced from the source PMF as

$$\lambda_{i,j,i',j'} = \mathbb{P}(R_t = r_{i,j} | R_{t+1} = r_{i',j'}) \tag{7.7}$$

$$= \mathbb{P}(S_t = a_i, \overline{L}_t = \overline{l}_{i,j} | \overline{B}_{t+1} = \overline{b}_{i',j'}) \tag{7.8}$$

$$= \begin{cases} \mathbb{P}(a_i) & \text{if } \overline{l}_{i,j} \sqsubset \overline{b}_{i',j'}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.9}$$

■

Let us denote $\Lambda = (\lambda_{i,j,i',j'})_{(i,j),(i',j')}$ the matrix of transition probabilities of the process $(Z_{t'})$. This matrix being known, it is possible to check whether the corresponding process is irreducible and aperiodic or not. Let us denote $\mathbb{I}$ the identity matrix which has the same dimension as Λ. If the matrix $\mathbb{I} - \Lambda$ is invertible, the solution of the linear system $\pi = \Lambda\pi$ is unique (see, e.g., [CT91]). In the following, we will assume that the VLRS and the source are defined so that the encoding process represented by $(Z_{t'})_{t'}$ is ergodic. If it is not the case, some rules can not be applied anymore when the sequence length becomes sufficiently high. This may occur when 1) some production rules of the VLRS are useless, 2) the source is singular ($\exists i / \ \mathbb{P}(a_i) = 0$) or 3) the VLRS does not define a valid prefix code.

The markov chain $(Z_{t'})$ being irreducible, the marginal probability distribution $\pi \triangleq \mathbb{P}(Z_{t'} = r_{i,j})$ is obtained from the transition matrix Λ as the unique solution of the equation $\pi = \Lambda\pi$ such that $||\pi||_1 = 1$. Therefore π is the normalized eigenvector associated with the eigenvalue 1. As t' grows to infinity (which requires that $t \to \infty$), the quantity $\delta(Z_{t'})$ is the expectation of the number of bits generated by a production rule. Given that $\delta(\mathbf{U})$ is a constant, from the Cesaro theorem, one can deduce the asymptotic value of the expected length as the sequence length increases, i.e. the quantity $\mathrm{EDL}_\infty(\mathcal{C})$.

Example 7.7: For the code $\mathcal{C}_4$, the transition matrix corresponding to the source PMF of Example 7.6 is

$$\begin{bmatrix} 0 & 0.7 & 0.7 & 0.7 \\ 0.7 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix},$$

which leads to $\mathbb{P}(R_t = r_{i,j}) = \{0.412, 0.288, 0.2, 0.1\}$. Finally, the expected length of this code is $EDL(\mathcal{C}_4) = 0.412 \times 0 + 0.288 \times 1 + 0.2 \times 3 + 0.1 \times 3 = 1.188$.

The expected length obtained in Example 7.7 is much closer to the entropy than the expected length obtained with Huffman codes. The expected number of bits required to code the symbol a_1 is less than 0.5 bit. One can also process the exact expected length of a VLRS for sequences of finite length. Indeed, the expectation of the number of termination bit(s) as well as the PMF $\mathbb{P}(R_t = r_{i,j}|t = L(\mathbf{S}))$ of the last rule can be obtained from the termination bit choice and from the source PMF The exact probability $\mathbb{P}(R_t = r_{i,j}|t = \tau)$ of having a given rule for a given instant τ can then be computed and subsequently one can deduce the expectation of the number of bits generated to encode the symbol S_τ.

A first-order approximation of this expected length may be obtained by assuming that $0s$ and $1s$ have the same probability, hence allowing to compute directly the probabilities $\mathbb{P}(R_t)$. For the code $\mathcal{C}_4$, this approximation leads to $\mathrm{EDL}(\mathcal{C}_4) = 1.25$.

7.5 Lexicographic code design

This section describes a VLRS construction method which allows to preserve the lexicographic order of the source alphabet in the binary coded representation. As a starting point, we assume that the Huffman code corresponding to the source PMF $\boldsymbol{\mu}$ is already known. The length of the Huffman codeword associated with the symbol a_i is denoted k_i. Let $k^+ = \max_i k_i$ denote the length of the longest codeword.

First, let us underline that the union $\bigcup_{i,j}\{\overline{b}_{i,j}\}$ of all the bit sequences $\overline{b}_{i,j}$ forms a Fixed Length Code (FLC) $\mathcal{F}$ of length k^+. $\mathcal{F}$ contains 2^{k^+} codewords. These code-

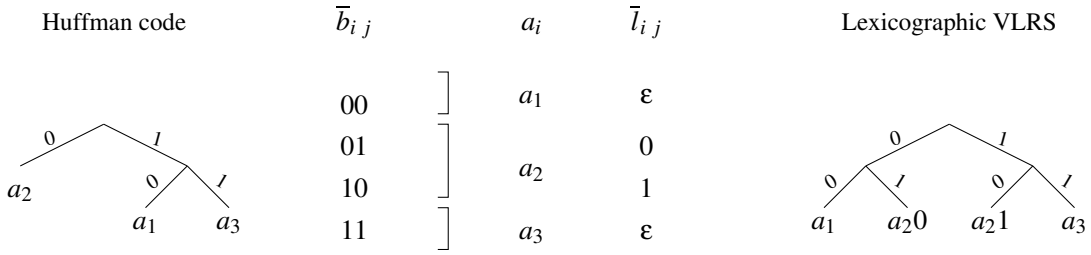


FIG. 7.3 – Construction of a lexicographic VLRS.

words will be assigned to production rules in the lexicographic order. Starting with the smaller symbol a_1, the algorithm proceeds as follows

1. $2^{k^+-k_i}$ rules are defined for the symbol a_i.
2. The left part of these rules are defined so that the set $\{\bar{l}_{i,j}\}_{j\in[1..|\mathcal{R}_i|]}$ forms a FLC of length $k^+ - k_i$. If $k_i = k^+$, this FLC only contains the element ε.
3. The $2^{k^+-k_i}$ smallest remaining codewords of $\mathcal{F}$, i.e. those which have not been assigned to previous symbols of $\mathcal{F}$, are then assigned to these production rules so that $\forall j, \bar{l}_{i,j} \leq \bar{l}_{i,j'} \Rightarrow \overline{b}_{i,j}\bar{l} \leq \overline{b}_{i,j'}$.
4. If $i = |\mathcal{A}|$, the construction procedure is completed. Otherwise the algorithm restarts at Step 1 with the symbol a_{i+1}.

By construction, the proposed algorithm leads to a VLRS with the lexicographic property and with the same compression efficiency as the code from which it is constructed. In some cases, the set of production rules generated in previous steps may be simplified according to property 7.4.

Example 7.8: Let us now assume that the source $\mathbf{S}$ *is memoryless with its PMF defined as* $\boldsymbol{\mu}_2 = \{0.2, 0.7, 0.1\}$. *Since* a_2 *has the highest probability, the Huffman code* $\mathcal{H}_2 = \{10, 0, 11\}$ *corresponding to this PMF is not lexicographic. The VLRS is constructed according to the proposed construction procedure, depicted in Fig. 7.3. For* $\mathcal{H}_2$*, we have* $k_2 = 1$ *and* $k_1 = k_3 = k^+ = 2$. *Hence* $\mathcal{F} = \{00, 01, 10, 11\}$. *Since* $k_1 = 2$*, only* 1 *production rule* $r_{1,1}$ *is assigned to the symbol* a_1 *and* $b_{1,1} = \varepsilon$*, which implies* $r_{1,1} : a_1 \rightarrow 00$. *The symbol* a_1 *is then assigned two production rules* $r_{2,1}$ *and* $r_{2,2}$ *as follows :*

$$r_{2,1} : a_2 0 \;\rightarrow\; 01$$
$$r_{2,2} : a_2 1 \;\rightarrow\; 10.$$

The construction algorithm finishes with the assignment of rule $r_{3,1} : a_3 \rightarrow 11$ *to symbol* a_3. *Finally, we obtain the code* $\mathcal{C}_3$ *proposed in Section 7.2, for which the EDL is equal to* 1.3 *and which has the lexicographic property. The Hu-Tucker code associated with this source is the code* $\mathcal{C}_1$ *proposed in Example 7.1 and its EDL is equal to* 1.8.

	Block length	EDL	Number of states
Hu-Tucker codes	1	1.900	2
(*generalized*)	2	1.270	8
	3	1.078	26
	4	0.998	80
Lexicographic VLRS	1	1.200	3
(*generalized*)	2	0.960	≤ 63

TAB. 7.1 – Proposed lexicographic construction method against Hu-Tucker codes : trade-off between the EDL and the number of states.

In Example 7.8, the corresponding FSMs do not have the same number of states (2 for the Hu-Tucker code *versus* 3 for the proposed VLRS). The EDL is compared in Table 7.1 with respect to the number of states in the decoding FSM for a 3-symbol source. The performance depicted are the ones of the generalized Hu-Tucker codes and the lexicographic VLRSs constructed from the generalized Huffman codes. Let us recall that, by generalized, we mean the Hu-Tucker and Huffman codes applied on the product alphabet $\mathcal{A}^n$. Such an alphabet contains $|A|^n$ elements. That example evidences that the Hu-Tucker codes are outperformed by the proposed VLRSs for this particular source for the trade-off between the compression efficiency and the number of states of the decoder. However, it is worth noticing that, even though the proposed construction allows to obtain lexicographic codes with the same compression efficiency as codes from which they are constructed, it does not construct, in general, the best lexicographic VLRSs from a compression efficiency point of view. One may find some lexicographic VLRS with lower EDL.

7.6 Mirror Code Design

The code design described in this section allows to obtain codes with marginal bit probabilities that are asymptotically equal to 0.5 as the sequence length increases. Let us consider a VLC code $\mathcal{H} = \{\overline{b}_{1,1}, \ldots, \overline{b}_{|\mathcal{A}|,1}\}$ and the code $\tilde{\mathcal{H}} = \{\tilde{\overline{b}}_{1,1}, \ldots, \tilde{\overline{b}}_{|\mathcal{A}|,1}\}$ defined so that each bit transition of the codetree characterizing $\tilde{\mathcal{H}}$ is the opposite value from the corresponding bit transition in $\mathcal{H}$, as depicted in Fig. 7.4.

The VLRS is obtained by putting together these two codes. The codes $\mathcal{H}$ and $\tilde{\mathcal{H}}$ are respectively used to define the two sets of $|\mathcal{A}|$ production rules forming the new VLRS as

$$\mathcal{M} = \begin{cases} \{a_i b_{i,1}^{L(\overline{b}_{i,1})} \rightarrow 0\,\overline{b}_{i,1}\}_{i\in[1..|\mathcal{A}|]} \\ \{a_i \tilde{b}_{i,1}^{L(\tilde{\overline{b}}_{i,1})} \rightarrow 1\,\tilde{\overline{b}}_{i,1}\}_{i\in[1..|\mathcal{A}|]}. \end{cases} \tag{7.10}$$

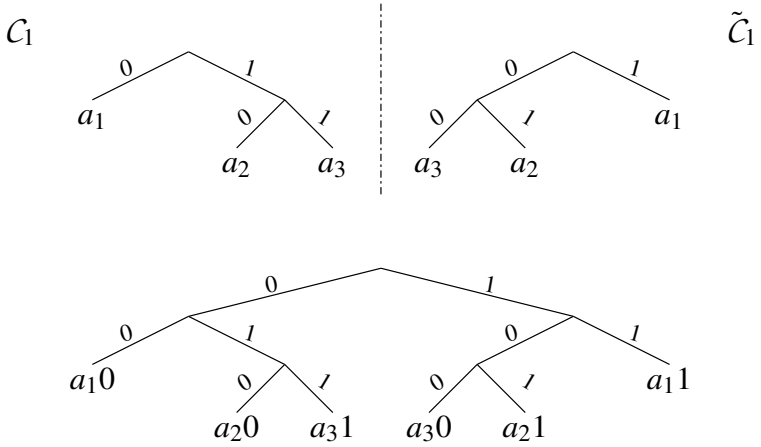


FIG. 7.4 – Primitive code $\mathcal{C}_1$, its opposite $\tilde{\mathcal{C}}_1$ and the resulting mirror VLRS.

Note that the production rules associated with codes $\mathcal{H}$ and $\tilde{\mathcal{H}}$ respectively define the subtrees corresponding to bit transitions 0 and 1.

Property 7.7: The VLRS $\mathcal{M}$ is a suffix-constrained code.

Proof: By construction. ■

Example 7.9: The construction associated with the code $\mathcal{C}_1$ leads to the following VLRS :

$$\underbrace{\left\{\begin{array}{lllll} r_{1,1}: & a_1 0 & \rightarrow & 00 \\ r_{2,1}: & a_2 0 & \rightarrow & 010 \\ r_{3,1}: & a_3 1 & \rightarrow & 011 \end{array}\right.}_{\text{obtained from } \mathcal{H}=\mathcal{C}_1}$$

$$\underbrace{\left\{\begin{array}{lllll} r_{1,2}: & a_1 1 & \rightarrow & 11 \\ r_{2,2}: & a_2 1 & \rightarrow & 101 \\ r_{3,2}: & a_3 0 & \rightarrow & 100. \end{array}\right.}_{\text{obtained from } \tilde{\mathcal{H}}=\tilde{\mathcal{C}}_1} \tag{7.11}$$

Property 7.8: $\forall n,\ \lim_{L(\mathbf{S})\rightarrow\infty} \mathbb{P}(E_n = 0) = 0.5$.

Proof: Let us consider a VLRS $\mathcal{M}$ constructed according to the previous guidelines. The notation $b_{i,j}$ refers to this VLRS (not to the VLC from which it is constructed). Let $f_t = \mathbb{P}(B_t^1 = 0)$ denote the marginal bit probability associated with the first bit

generated by a given production rule. Since the VLRS is constructed from a VLC, we have $\forall i, j,\ \delta(r_{i,j}) \geq 1$, which means that every rule produces at least one bit. The value f_t can be written as

$$f_t = \sum_{i\in[1..|\mathcal{A}|],j\in[1..2]} \mathbb{P}(R_t = r_{i,j}, B_t^1 = 0) \tag{7.12}$$

$$= \sum_{i\in[1..|\mathcal{A}|],j=1} \mathbb{P}(S_t = a_i, B_{t+1}^1 = l_{i,1}^{L(l_{i,1})}) \tag{7.13}$$

$$= \sum_{i\in[1..|\mathcal{A}|],j=1} \mathbb{P}(S_t = a_i, L_t^{L(L_t)} = 0)\, f_{t+1}$$

$$+ \sum_{i\in[1..|\mathcal{A}|],j=1} \mathbb{P}(S_t = a_i, L_t^{L(L_t)} = 1)\,(1 - f_{t+1}). \tag{7.14}$$

Let

$$\alpha \triangleq \sum_{a_i\in\mathcal{A}} \mathbb{P}(a_i, l_{i,1}^{L(\bar{l}_{i,1})} = 0). \tag{7.15}$$

This entity corresponds to the sum of the probabilities of the symbols to which a codeword ending with 0 has been assigned. Note that $\forall i,\ \mathbb{P}(a_i) > 0 \Rightarrow 0 < \alpha < 1$. Inserting this entity in Eqn. 7.14, we obtain

$$f_t = \alpha\, f_{t+1} + (1 - \alpha)(1 - f_{t+1}). \tag{7.16}$$

We can now study the asymptotic behavior of this sequence as $t' = L(\mathbf{S}) - t + 1$ tends to $+\infty$ (note that $f_{L(\mathbf{S})}$ is a constant). The absolute value of the derivative of the function $g(x) = \alpha\, x + (1 - \alpha)(1 - x)$ is stricly lower than 1 when $0 < \alpha < 1$. Consequently, the fixed-point theorem applies and the sequence $f_{L(\mathbf{S})}, f_{L(\mathbf{S})-1}, \ldots, f_{t'}$ converges to the solution of $x = g(x)$, which is 0.5. Subsequently, $\forall i$, opposite codewords $\overline{b}_{i,1}$ and $\overline{b}_{i,2}$ are equiprobable,

Since $\mathcal{M}$ is a suffix-constrained code, a bit produced by a production rule R_t is not modified by any subsequent production rule. Hence, the bits $b_{i,j}$ produced by the rules of the code $\mathcal{M}$ actually correspond to the bits forming the bitstream $\mathbf{S}$. This concludes the proof. ■

7.7 Soft decoding and simulation results

The amenability of the mirror codes to improve the performance of soft decoding has been assessed by simulations. For this purpose, we have considered the RLVC $\mathcal{C}_{12}$ of [TWM95], defined as

$$\mathcal{C}_{12} = \{00, 11, 010, 101, 0110\}.$$

The corresponding mirror code is then defined by the set of rules that follows.

$$\begin{cases} a_1 0 & \rightarrow & 0 00 \\ a_1 1 & \rightarrow & 1 11 \\ a_2 0 & \rightarrow & 1 00 \\ a_2 1 & \rightarrow & 0 11 \\ a_3 0 & \rightarrow & 0 010 \\ a_3 1 & \rightarrow & 1 101 \\ a_4 0 & \rightarrow & 1 010 \\ a_4 1 & \rightarrow & 0 101 \\ a_5 0 & \rightarrow & 0 0110 \\ a_5 1 & \rightarrow & 1 1001. \end{cases}$$

Since this code is reversible, this set of rules can also be considered by reversing both the codewords and by interverting the symbol and the bit in the left part of the rule. It amounts to considering rules of the form $\bar{l}a \rightarrow \bar{b}$ instead :

$$\begin{cases} 0 a_1 & \rightarrow & 0 00 \\ 0 a_2 & \rightarrow & 0 01 \\ 0 a_3 & \rightarrow & 0 100 \\ 0 a_4 & \rightarrow & 0 101 \\ 0 a_5 & \rightarrow & 0 1100 \\ 1 a_1 & \rightarrow & 1 11 \\ 1 a_2 & \rightarrow & 1 10 \\ 1 a_3 & \rightarrow & 1 011 \\ 1 a_4 & \rightarrow & 1 010 \\ 1 a_5 & \rightarrow & 1 0011. \end{cases}$$

Since the original code was suffix free, the VLC associated with the mirrored code is also suffix free. Consequently, the code defined by the inverted rules is prefix free. Moreover, by construction the bit conditionning the choice of the rule is not modified by a subsequent rule. The set of codewords associated with a given conditioning bit also forms a prefix free code. As a consequence, a sequence encoded with a mirrored code can be instantaneoulsy decoded in the backward direction as well. The states of the corresponding decoder then correspond to the internal nodes of the codetree formed by the left part of the rule. In our example, these internal nodes correspond to the code prefixes

$$\{\varepsilon, 0, 1, 00, 01, 10, 11, 010, 0110, 101, 1001\}.$$

It is of interest because it is then possible to apply either the BCJR algorithm or the Viterbi algorithm on the state model composed of these states [4]. Note that this model is similar to the one proposed by Balakirsky for VLCs in [Bal97].

[4] It can also be seen as a couple $(bit, \mathcal{N})$ comprising the last bit (in the backward direction) and an internal node of a new codetree with $|\mathcal{A}|$ leaves deduced from the set of rules, here defined as $\{00, 01, 100, 101, 1100\}$.

This soft decoding on this state model has then been applied for two sources distribution. The first distribution is quite balanced and given by

$$\mu_1 = \{0.4, 0.2, 0.2, 0.1, 0.1\}.$$

Its marginal bit probability is equal to 0.6. The second source is defined as

$$\mu_1 = \{0.9, 0.025, 0.025, 0.025, 0.025\}$$

with $\mathbb{P}(0) = 0.917$.

The proposed code has been compared against the original code and depicted in Fig. 7.5 and 7.6. The simulations have been performed assuming an AWGN channel together with a BPSK modulation. As expected, the improvement is important for the unbalanced source only. Note however that this improvement is free since the expected length of the code is identical to the one of the original code.

7.8 Concluding remarks

The formalism based on variable length re-writing rules offers degrees of freedom allowing the design of codes with interesting properties, in particular for Joint source channel/coding. In this chapter, VLRSs have been defined by rules of the form $a\bar{l} \rightarrow \bar{b}$. The same kind of analysis applies if rules of the form $\bar{l}a \rightarrow \bar{b}$ are considered instead, as suggested in Section 7.7.

The soft decoding performance obtained with these codes illustrates the interest of considering some source codes with better statistical properties. Note that these VLRSs can be easily decoded jointly with a convolutional code in an iterative decoding setup, in the spirit of the work of [BH00b].

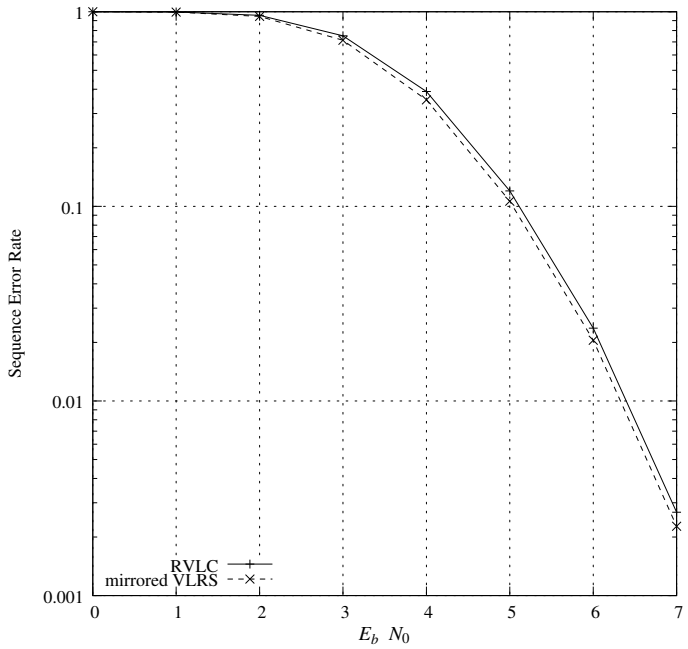


FIG. 7.5 – Soft Decoding of Mirrored VLRS : the balanced case.

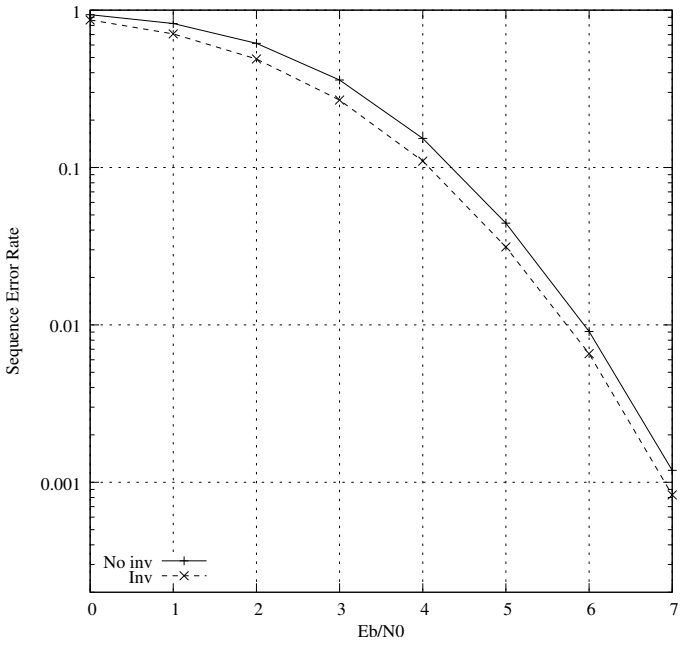


FIG. 7.6 – Soft Decoding of Mirrored VLRS : the unbalanced case.

Glossaire/Glossary

English	*French* (if used)	
AWGN		Additive white gaussian noise channel
BC		Bitstream construction
BER		Bit error rate
BPSK		Binary phase shift keying
BSC		Binary symmetric channel
CA		Codeword assignment
CABAC		Context adaptative binary arithmetic coder
CAVLC		Context adapative variable length codes
CC		Channel Codes
COVQ		Channel optimized vector quantization
DMC		Discrete memoryless channel
EDL		Expected description length
EREC		Error resilient entropy coder
FLC	CLF	Fixed length code
FSM		Finite state machine
GCD		Greatest common divisor
IA		Index assignment
LDPC		Low density parity check codes
MAP	MAP	Maximum a posteriori
MEPL		Mean error propagation length
MMSE	MEQM	Minimum of mean square error
MPM	MMP	Maximum posterior marginals
MS		Markov source
MSE	EQM	Mean square error
NLD		Normalized Levenshtein distance
PMF		Probability mass function
PSNR		Peak signal to noise ratio
RCPC		Rate compatible convolutional codes
RSCC		Recursive systematic convolutional codes
RLE		Run-length encoding
RVLC	CLVR	Reversible variable length codes
SA		Simulated annealing
SC		Source coder
SER	TES	Symbol error rate
SISO		Soft input soft output
SQER		Sequence error rate
SNR		Signal to noise ratio
UEP		Unequal error protection
VEPL		Variance of the error propagation length
VLC	CLV	Variable length code
VLRS		Variable length re-writing system

Bibliographie

[Abr63] N. Abramson. *Information theory and coding*. McGraw-Hill, New York, 1963.

[Ari72] S. Arimoto. An algorithm for computing the capacity of arbitrary DMCs. *IEEE Trans. Inform. Theory*, 18 :11–20, 1972.

[Bal97] V. B. Balakirsky. Joint source-channel coding with variable length codes. In *Proc. Intl. Conf. Inform. Theory*, 1997. p.419.

[BCJR74] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory*, 20 :284–287, Mar. 1974.

[BF93] V. Buttigieg and P.G. Farrell. A maximum likelihood decoding algorithm for variable-length error-correcting codes. In *Proc. 5th Bangor Symposium on Communications*, pages 56–59, June 1993.

[BGP93] C. Berrou, A. Glavieux, and Thitimajshima P. Near Shannon limit error-correcting coding and decoding : Turbo codes. In *Proc. Intl. Conf. Commun.*, pages 1064–1070, 1993.

[BH00a] R. Bauer and J. Hagenauer. Iterative source-channel decoding based on a trellis representation for variable length codes. In *Proc. Intl. Conf. Inform. Theory*, page 238, June 2000.

[BH00b] R. Bauer and J. Hagenauer. Iterative source-channel decoding using reversible variable length codes. In *Proc. Data Compression Conf.*, pages 93–102, Mar. 2000.

[BH00c] R. Bauer and J. Hagenauer. Symbol by symbol MAP decoding of variable length codes. In *Proc. 3rd ITG Conf. Source and Channel Coding, Munich, Germany, Jan. 17.-19. 2000*, 2000.

[BH00d] R. Bauer and J. Hagenauer. Turbo FEC/VLC decoding and its application to text compression. In *Proc. Conf. Inform. Theory and Systems*, pages WA6.6–WA6.11, Mar. 2000.

[BK93] A. Bookstein and S.T. Klein. Is Huffman coding dead ? *Computing*, 50 :279–296, 1993.

[BKK01] M. Bystrom, S. Kaiser, and A. Kopansky. Soft source decoding with applications. *IEEE Trans. Circuits Syst. Video Technol.*, 11(10) :1108–1120, Oct. 2001.

[Bla72] R. Blahut. Computation of channel capacity and rate distortion functions. *IEEE Trans. Inform. Theory*, 18 :460–473, Mar. 1972.

[Bla79] R.E. Blahut. Transform techniques for error control codes. *IBM J.RES. DEVELOP*, 23(3), May 1979.

[Bla05] I. F. Blake. Book review : Error control coding. *IEEE Trans. Inform. Theory*, 51(4) :1616–1617, May 2005.

[But95] V. Buttigieg. *Variable-length error-correcting codes*. PhD thesis, University of Manchester, 1995.

[CGV88] R.M. Capocelli, L. Gargano, and U. Vaccaro. On the characterization of statistically synchronizable variable length codes. *IEEE Trans. Inform. Theory*, 34(4) :817–825, Jul. 1988.

[CP05] S. Cho and W. A. Pearlman. Multilayered protection of embedded video bitstreams over binary symmetric and packet erasure channels. *J. Visual Communication and Image Representation*, 16 :359–378, June 2005.

[CSGV92] R.M. Capocelli, A.A.D. Santis, L. Gargano, and U. Vaccaro. On the construction of statistically synchronizable variable length codes. *IEEE Trans. Inform. Theory*, 38(2) :407–414, Mar. 1992.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[CZ96] G. Cheung and A. Zakhor. Joint source-channel coding of scalable video over noisy channels. In *Proc. Intl. Conf. Image Processing*, pages 767–770, 1996.

[DG81] J. G. Dunham and R. M. Gray. Joint source and noisy channel trellis encoding. *IEEE Trans. Inform. Theory*, 27(4) :516–519, July 1981.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1 :269–271, 1959.

[DS98] N. Demir and K. Sayood. Joint source-channel coding for variable length codes. In *Proc. Data Compression Conf.*, pages 139–148, Mar. 1998.

[DS04] M. Drmota and W. Szpankowski. Variable-to-variable length codes with small redundancy rates. In *Proc. of the 2002 IEEE International Symposium on Information Theory*, June 2004.

[DW05] S. Dumitrescu and X. Wu. On the complexity of joint source-channel decoding of Markov sequences over memoryless channels. In *Proc. Intl. Conf. Inform. Theory*, Sept. 2005.

[EM02] Y. Ephraim and N. Merhav. Hidden Markov process. *IEEE Trans. Inform. Theory*, 48(6) :1518–1569, June 2002.

[Eri05] T. Eriksson. *Trellis Source Coding Methods for Low Rates and Short Blocks*. PhD thesis, Lund University, May 2005.

[Far90] N. Farvardin. A study of vector quantization for noisy channels. *IEEE Trans. Inform. Theory*, 36(4) :799–809, July 1990.

[FR84] T.J. Ferguson and J. H. Rabinowitz. Self-synchronizing Huffman codes. *IEEE Trans. Inform. Theory*, 30(4) :687–693, July 1984.

[FV91] N. Farvardin and V. Vaishampayan. On the performance and complexity of channel-optimized vector quantizers. *IEEE Trans. Inform. Theory*, 37(1) :155–160, Jan. 1991.

[Gab01] A. Gabay. *Codage conjoint source canal-Applications aux transmissions d'images par satellites*. PhD thesis, Thèse, Ecole Nationale Supérieure des Télécommunications, Jan. 2001.

[Gal62] R. G. Gallager. Low density parity check codes. *Trans. IRE Prof. Group on Inform. Theory*, 8 :2l–28, Jan. 1962. *http ://web.mit.edu/gallager/www/notes/pubs.pdf*.

[Gal94] R. G. Gallager. The Arimoto-Blahut algorithm for finding channel capacity, Mar. 1994. technical note, *http ://web.mit.edu/gallager/www/notes/notes4.pdf*.

[GCS00] L. Guivarch, J.C. Carlach, and P. Siohan. Joint source-channel soft decoding of Huffman sources with turbo-codes. In *Proc. Data Compression Conf.*, pages 83–92, Mar. 2000.

[GFGR01] A. Guyader, E. Fabre, C. Guillemot, and M. Robert. Joint source-channel turbo decoding of entropy coded sources. *IEEE J. Select. Areas Commun.*, 19(9) :1680–1696, Sept. 2001.

[GG03] T. Guionnet and C. Guillemot. Soft decoding and synchronization of arithmetic codes : Application to image transmission over noisy channels. *IEEE transactions on Image Processing*, 12(12) :1599–1609, Dec. 2003.

[GG04] T. Guionnet and C. Guillemot. Soft and joint source-channel decoding of quasi-arithmetic codes. *EURASIP Journal on Applied Signal Processing*, 2004(3) :393–411, Mar. 2004.

[GHSW87] A.A El Gamal, L. Hemachandra, I. Shperling, and V. Wei. Using simulated annealing to design good codes. *IEEE Trans. Inform. Theory*, 33(1) :116–123, Jan. 1987.

[GMP] The gmp library. *http ://www.swox.com/gmp/*.

[GN98] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44 :2325–2384, Oct. 1998.

[Gol66] S.W. Golomb. Run-length encodings. *IEEE Trans. Inform. Theory*, 12(3) :399–401, 1966.

[Gor94] M. J. Gormish. *Source Coding with Channel, Distortion and Complexity Constraints*. PhD thesis, Stanford, Mar. 1994.

[Gra90] R. M. Gray. *Entropy and Information Theory*. Springer-Verlag, 1990.

[Guy02] A. Guyader. *Contribution aux algorithmes de décodage dans les codes graphiques*. PhD thesis, University of Rennes 1, 2002.

[Hag88] J. Hagenauer. Rate-compatible punctured convolutional codes and their applications. *IEEE Trans. Commun.*, 36(4) :389–400, Apr. 1988.

[HT71] T. C. Hu and A. C. Tucker. Optimal computer search trees and variable length alphabetic codes. *SIAM J. Appl. Math.*, 21(4) :514–532, 1971.

[Huf52] D. Huffman. A method for the construction of minimum redundancy codes. In *Proc. of the IRE*, volume 40, pages 1098–1101, 1952.

[HV92] P. G. Howard and J. S. Vitter. Practical implementations of arithmetic coding. *Image and Text compression*, pages 85–112, 1992.

[JCS05] M. Jeanne, J.C. Carlach, and P. Siohan. Joint source-channel decoding of variable length codes for convolutional codes and turbo codes. *IEEE Trans. Commun.*, 53(1) :10–15, Jan. 2005.

[Jég] H. Jégou. Source code : C++ implementation of the for proposed algorithms, *http ://www.irisa.fr/temics/Equipe/Jegou/src/source_code.php*.

[JG03a] H. Jégou and C. Guillemot. Error-resilient binary multiplexed source codes. In *Proc. Intl. Conf. Acc. Speech Signal Processing*, Apr. 2003.

[JG03b] H. Jégou and C. Guillemot. First-order multiplexed codes : performance bounds, design and decoding algorithms. *IEEE Trans. Signal Processing*, 2003. *Accepted*.

[JG03c] H. Jégou and C. Guillemot. A new class of codes for transmission of heterogeneous data : Multiplexed codes, Aug. 2003. INRIA Research Report 4922.

[JG03d] H. Jégou and C. Guillemot. Source multiplexed codes for error-prone channels. In *Proc. Intl. Conf. Commun.*, May 2003.

[JG04a] H. Jégou and C. Guillemot. Error-resilient and progressive transmission and decoding of compressed images. In *Proc. WIAMIS'2004*, Apr. 2004. Lisboa.

[JG04b] H. Jégou and C. Guillemot. First-order multiplexed source codes for error-resilient entropy coding. In *Proc. Intl. Conf. Inform. Theory*, June 2004. Chicago.

[JG04c] H. Jégou and C. Guillemot. Suffix-constrained codes for progressive and robust data compression : self-multiplexed codes. In *Proc. EUSIPCO'2004*, Sept. 2004. Vienna.

[JG05a] H. Jégou and C. Guillemot. Codage entropique à base de règles de réécriture. In *GRETSI*, Sept. 2005. Louvain-La-Neuve, Belgium.

[JG05b] H. Jégou and C. Guillemot. Entropy coding with variable length rewriting systems. In *Proc. Intl. Conf. Inform. Theory*, Sept. 2005. Adelaide, Australia.

[JG05c] H. Jégou and C. Guillemot. Progressive and error-resilient strategies for VLC encoded signals over noisy channels. *EURASIP Journal of Applied Signal Processing*, 2005. *Accepted*.

[JG05d] H. Jégou and C. Guillemot. Robust multiplexed codes for compression of heterogeneous data. *IEEE Trans. Inform. Theory*, pages 1393–1407, Apr. 2005.

[JMG05a] H. Jégou, S. Malinowski, and C. Guillemot. Décodage conjoint source/canal par aggrégation d'états et décodage multi-treillis. In *Compression et représentation des signaux audiovisuels*, Rennes, France, Nov. 2005.

[JMG05b] H. Jégou, S. Malinowski, and C. Guillemot. Trellis state aggregation for soft decoding of variable length codes. In *IEEE Workshop on Signal Processing Systems*, Athens, Greece, Nov. 2005.

[JV04] X. Jaspar and L. Vandendorpe. New iterative decoding of variable length codes with turbo codes. In *Proc. Intl. Conf. Commun.*, June 2004. Paris, France.

[JV05] X. Jaspar and L. Vandendorpe. Performance and convergence analysis of joint source-channel turbo schemes with variable length codes. In *Proc. Intl. Conf. Acc. Speech Signal Processing*, volume 3, pages 485–488, Philadelphia, USA, Mar. 2005.

[KDKM00] A. Kiely, S. Dolinar, M. Klimesh, and A. Matache. Error containment in compressed data using sync markers. In *Proc. Intl. Conf. Inform. Theory*, page 428, June 2000.

[Kho72] G. L. Khodak. Bounds of redundancy estimates for word-based encoding of sequences produced by a Bernouilli source. *Problemy Peradachi Informacii*, 8 :21–32, 1972.

[KN00] N. Kashyap and D. L. Neuhoff. Data synchronisation with timing. In *Proc. Intl. Conf. Inform. Theory*, page 427, June 2000.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, Jan. 1963. *http ://cr.yp.to/bib/1963/karatsuba.html*.

[KT03] J. Kliewer and R. Thobaben. Parallel concatenated joint source-channel coding. *Electronics Letters*, 39(23) :1664–1666, Nov. 2003.

[KT05] J. Kliewer and R. Thobaben. Iterative joint source-channel decoding of variable-length codes using residual source redundancy. *IEEE Trans. Wireless Commun.*, 4(3) :919–929, May 2005.

[KY00] J. Kieffer and E.-H Yang. Grammar-based codes : a new class of universal lossless source codes. *IEEE Trans. Inform. Theory*, 46 :737–754, 2000.

[LC04] S. Lin and D. Costello. *Error Control Coding*. Prentice-Hall, 2004. 2nd edition.

[Lev66] Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10 :707–710, 1966.

[LH90] L. L. Larmore and D. S. Hirschberg. A fast algorithm for optimal length-limited Huffman codes. *J. Assoc. Computing Machinery*, 37 :464–473, 1990.

[LKH+00] A. H. Li, S. Kittitornkun, Y. Hu, D Park, and J.Villasenor. Data partitioning and reversible variable length codes for robust video communications. In *Proc. Data Compression Conf.*, pages 460–469, Mar. 2000.

[LR92] W.M. Lam and A.R. Reibman. Self-synchronizing variable-length codes for image transmission. In *Proc. Intl. Conf. Acc. Speech Signal Processing*, volume Mar., pages 477–480, Sept. 1992.

[LS03] S. Lonardi and W. Szpankowski. Joint source-channel LZ'77 coding. In *Proc. Data Compression Conf.*, pages 273–282, 2003.

[Mac03] D. J.C. MacKay. *Information Theory, Inference and Learning algorithms*. Cambridge University Press, 2003.

[Mal05] S. Malinowski. Décodage souple des codes à longueur variable. Master's thesis, Ecole Nat. Supérieur des Télécommunications de Bretagne, 2005.

[MF98a] A.H. Murad and T.E. Fuja. Exploiting the residual redundancy in motion estimation vectors to improve the quality of compressed video transmitted over noisy channels. In *Proc. Intl. Conf. Image Processing*, Oct. 1998. Chicago.

[MF98b] A.H. Murad and T.E. Fuja. Joint source-channel decoding of variable length encoded sources. In *Proc. Inform. Theory Workshop*, pages 94–95, June 1998.

[MF99] A.H. Murad and T.E. Fuja. Robust transmission of variable-length encoded sources. In *Proc. IEEE Wireless Communications and Networking Conf.*, pages 964–968, Sept. 1999.

[MF00] A. H. Murad and T. E. Fuja. Improving the performance of variable-length encoded systems through cooperation between source and channel decoders. In *Proc. Intl. Conf. Inform. Theory*, page 264, June 2000.

[MJG05] S. Malinowski, H. Jégou, and C. Guillemot. Synchronization recovery and state model reduction for soft decoding of variable length codes. *IEEE Trans. Inform. Theory*, Sept. 2005. submitted.

[MKLKD05] G. R. Mohammad-Khani, C. M. Lee, M. Kieffer, and P. Duhamel. Treillis à complexité réduite pour le décodage de codes à longueur variable. In *Actes de GRETSI*, 2005.

[ML87] M. E. Monaca and J. M. Lawler. Corrections and additions to "error recovery for variable length codes". *IEEE Trans. Inform. Theory*, 33(3) :454–456, May 1987.

[MP98] D. J. Miller and M. Park. A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden markov models. *IEEE Trans. Commun.*, 46(2) :222–231, 1998.

[MR85] J.C. Maxted and J.P. Robinson. Error recovery for variables length codes. *IEEE Trans. Inform. Theory*, 31(6) :794–801, Nov. 1985.

[MSW03] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. Circuits Syst.*, 13(7), July 2003.

[ND03] H. N'guyen and P. Duhamel. Estimation of redundancy in compressed image and video data for joint source/channel decoding. In *Globecom*, pages 2198–2202, 2003.

[Pas76] R. C. Pasco. *Source Coding Algorithms for Fast Data Compression*. PhD thesis, Stanford University, CA, may 1976.

[PM98] M. Park and D. J. Miller. Decoding entropy-coded symbols over noisy channels using discrete HMMs. In *Conf. In Information Sciences and Systems*, Mar. 1998. Princeton.

[PSA98] R. Pyndiah, B. Solaiman, and O. Aitsab. Optimized source coding and coded modulation for digital transmission of still images on a noisy channel. In *Proc. Intl. Conf. Image Processing*, volume 2, pages 142–146, 1998.

[PSH00] B.D. Pettijohn, K. Sayood, and M.W. Hoffman. Joint source/channel coding using arithmetic codes. In *Proc. Data Compression Conf.*, Mar. 2000. Snowbird, Utah.

[PSR04] S. Perkins, D.H. Smith, and A. Ryley. Robust data compression : consistency checking in the synchronization of variable length codes. *The Computer Journal*, 47(3) :309–319, 2004.

[Pyn98] R. Pyndiah. Near optimum decoding of product codes : block turbo codes. *IEEE Trans. Commun.*, 46(8) :1003–1010, Aug. 1998.

[Ris76] J.J. Rissanen. Generalized Kraft inequality and arithmetic coding. *J. Research and Development*, 20(3) :198–203, 1976.

[RK96] D.W. Redmill and N.G. Kingsbury. The EREC : An error resilient technique for coding variable-length blocks of data. *IEEE Trans. Image Processing*, 5 :565–574, Apr. 1996.

[RM95] M. J. Ruf and J. W. Modestino. Rate-distortion performance for joint source and channel coding of images. *Proc. Intl. Conf. Image Processing*, 2 :77–80, Oct. 1995.

[Rob01] C. P. Robert. *The Bayesian Choice. From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics, 2001.

[Say99] J. Sayir. Arithmetic coding for noisy channels,. In *IEEE Information Theory Workshop*, June 1999. (Kruger Park, South Africa.

[Say00] J. Sayir. Iterating the Arimoto-Blahut algorithm for faster convergence. In *Proc. Intl. Conf. Inform. Theory*, page 235, June 2000.

[SB91] K. Sayood and J.C. Borkenhagen. Use of residual redundancy in the design of joint source/channel coders. *IEEE Trans. Commun.*, 39(6) :838–846, June 1991.

[SD95] P. F. Swaszek and P. DiCicco. More on the error recovery for variable length codes. *IEEE Trans. Inform. Theory*, 41(6) :2064–2071, Nov. 1995.

[SG97] S. A. Savari and R. G. Gallager. Generalized Tunstall codes for sources with memory. *IEEE Trans. Inform. Theory*, 43(2) :658–668, Mar. 1997.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J*, 27 :379–423 623–656, 1948.

[SOD00] K. Sayood, H. H. Otu, and N. Demir. Joint source/channel coding for variable length codes. *IEEE Trans. Commun.*, 48(5) :787–794, May 2000.

[SS02] S. A. Savari and W. Szpankowski. On the analysis of variable-to-variable length codes. In *Proc. of the 2002 IEEE International Symposium on Information Theory*, June 2002.

[SV98] K.P. Subbalakshmi and J. Vaisey. Optimal decoding of entropy coded memoryless sources over noisy channels. In *Proc. Data Compression Conf.*, page 573, Mar. 1998.

[SV99] K.P. Subbalakshmi and J. Vaisey. Joint source-channel decoding of entropy coded markov sources over binary symmetric channels. In *Proc. Intl. Conf. Commun.*, volume 1, pages 446–450, 1999.

[Szp00] W. Szpankowski. Asymptotic average redundancy of Huffman (and Shannon-Fano) block codes. In *Proc. Intl. Conf. Inform. Theory*, page 370, June 2000.

[Tau99] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Trans. Image Processing*, 9(7) :1158–1170, Sept. 1999.

[TF92] N. Tanabe and N. Farvardin. Subband image coding using entropy-coded quantization over noisy channels. *IEEE J. Select. Areas Commun.*, 10(5) :926–942, June 1992.

[Tit97] M. R. Titchener. The synchronization of variable-length codes. *IEEE Trans. Inform. Theory*, 43(2) :683–691, 1997.

[Tja00] T. Tjalkens. The complexity of minimum redundancy coding. In *Proc. Intl. Conf. Inform. Theory*, page 373, June 2000.

[TK03] R. Thobaben and J. Kliewer. Robust decoding of variable length encoded Markov sources using a three-dimensional trellis. *IEEE Trans. Commun.*, 7(7) :787–794, July 2003.

[Tun67] B.P. Tunstall. Synthesis of noiseless compression codes. *Ph.D Dissertation, Georgia Institute of Technology, Atlanta*, 1967.

[TW01] C.-W. Tsai and J.-L. Wu. On constructing the Huffman-code-based reversible variable-length codes. *IEEE Trans. Commun.*, 49(9) :1506–1509, Sept. 2001.

[TWM94] Y. Takishima, M. Wada, and H. Murakami. Error states and synchronization recovery for variable length codes. *IEEE Trans. Commun.*, 42(2/3/4) :783–792, Feb./Mar./Apr. 1994.

[TWM95] Y. Takishima, M. Wada, and H. Murakami. Reversible variable length codes. *IEEE Trans. Commun.*, 43(2/3/4) :158–162, Feb. 1995.

[Vit67] A. Viterbi. Error bounds for convolution codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13(2) :260–269, Apr. 1967.

[Voo74] D. C. Van Voorhis. Constructing codes with bounded codeword lengths. *IEEE Trans. Inform. Theory*, 20(2) :288–290, Mar. 1974.

[Wei03] C. Weidmann. Reduced-complexity soft-in-soft-out decoding of variable length codes. In *Proc. Intl. Conf. Inform. Theory*, July 2003. Yokohama, Japan.

[WF74] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1) :168–173, 1974.

[WK99] S. Whitehouse and N. Kingsbury. Enhancements to the error resilient entropy code. In *IEEE MMSP'98*, Dec. 1999. Los Angeles.

[WS80] V.K.W. Wei and R.A. Scholtz. On the characterization of statistically synchronizable codes. *IEEE Trans. Inform. Theory*, 26(6) :733–735, Nov. 1980.

[WS95] Ja-Ling Wu and Jiun Shiu. Discrete cosine transform in error control coding. *IEEE Trans. on Communications*, May 1995. 47(4) :1065 :1075.

[WV98] J. Wen and J. D. Villasenor. Reversible variable length codes for efficient and robust image and video coding. In *Proc. Data Compression Conf.*, pages 471–480, Apr. 1998.

[Xia03] W. Xiang. *Joint Source-Channel Coding for Image Transmission and Related Topics*. PhD thesis, University of South Australia, Dec. 2003.

[YY02] C. Ye and R. W. Yeung. A simple upper bound on the redundancy of Huffman codes. *IEEE Trans. Inform. Theory*, 48(7) :2132–2138, July 2002.

[ZA02] G.-C Zhu and F. Alajaji. Turbo codes for nonuniform memoryless sources over noisy channels. *IEEE Commun. Lett.*, 6 :1253–1262, Feb. 2002.

[ZABM04] G.-C Zhu, F. Alajaji, J. Bajcsy, and P. Mitran. Transmission of nonuniform memoryless sources via nonsystematic turbo codes. *IEEE Trans. Commun.*, 8(52) :1344–1354, Aug. 2004.

[ZG90] K. Zeger and A. Gersho. Pseudo-Gray coding. *IEEE Trans. Commun.*, 38(12) :2147–2158, Dec. 1990.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for data compression. *IEEE Trans. Inform. Theory*, 23(3) :337–343, 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory*, 24(5) :530–343, 1978.

[Zur94] D. Zuras. More on squaring and multiplying large integers. *IEEE Trans. Comput.*, 43(8) :899–908, Aug. 1994.

[ZZ02] G. Zhou and Z. Zhang. Synchronization recovery of variable length codes. *IEEE Trans. Inform. Theory*, 48(1) :219–227, Jan. 2002.

Table des figures

Résumé

Cette thèse propose des codes robustes et des codes conjoints source/canal pour transmettre des signaux multimédia sur des canaux bruités. Nous proposons des codes entropiques offrant une résistance intrinsèque aux données prioritaires. Ces codes sont étendus pour exploiter la dépendance temporelle du signal.

Un nouveau modèle d'état est ensuite proposé et analysé pour le décodage souple de codes à longueur variable avec une contrainte de longueur. Il permet de régler finement le compromis performance de décodage/complexité.

Nous proposons également de séparer, au niveau du codage entropique, les étapes de production des mots de codes et de paquétisation. Différentes stratégies de construction de train binaire sont alors proposées.

Enfin, la représentation en arbre binaire des codes entropiques est étendue en considérant des règles de ré-écriture. Cela permet en particulier d'obtenir des codes qui offrent des meilleures performances en décodage souple.

Abstract

Some new error-resilient source coding and joint source/channel coding techniques are proposed for the transmission of multimedia sources over error-prone channels. First, we introduce a class of entropy codes providing unequal error-resilience, i.e. providing some protection to the most sensitive information. These codes are then extended to exploit the temporal dependencies.

A new state model based on the aggregation of some states of the trellis is then proposed and analyzed for soft source decoding of variable length codes with a length constraint. It allows the weighting of the compromise between the estimation accuracy and the decoding complexity.

Next, some paquetization methods are proposed to reduce the error propagation phenomenon of variable length codes.

Finally, some re-writing rules are proposed to extend the binary codetree representation of entropy codes. The proposed representation allows in particular the design of codes with improved soft decoding performances.