ETUDE COMPARATIVE DES SYSTEMES DE COMPRESSION DE DONNEES SANS PERTE D'INFORMATION ET CODAGE ARITHMETIQUE

G. Furlan, C. Galand, E. Lançon

Centre d'Etudes et Recherche
IBM France
06610 La Gaude

RESUME

Les approches classiques de la compression de données sans perte d'information sont basées sur un découplage de la modélisation de la source et du codage. La modélisation statistique avec codage par la méthode d'Huffman est la combinaison la plus utilisée. On considère alors soit un modèle fixe (ce qui a l'avantage d'un codage simple passe mais qui ne peut être efficace pour des sources de types variés), soit un modèle adapté à chaque fichier mais qui nécessite une "double passe". L'algorithme "simple passe" de fragmentation croissante de Ziv-Lempel permet d'obtenir des taux de compression importants. Cependant, cet algorithme n'est efficace que pour de gros fichiers corrélés sur une seule dimension.

Afin de pallier ces divers inconvénients, l'algorithme "du contexte" permettant une modélisation markovienne adaptative de la source a été présenté par Rissanen en 1983. Avec sa modélisation en arbre et une gestion originale du modèle basée sur le principe de la longueur de description minimale, cet algorithme peut s'appliquer à des sources corrélées sur plusieurs dimensions, telles que les images. La partie codage est assurée par un code arithmétique multi-niveaux sans multiplication, qui a l'avantage, par rapport aux codes "préfix", de retrouver parfaitement l'entropie quelque soit la statistique et d'être parfaitement adaptatif en cas de modification du modèle.

L'article est organisé en trois parties : nous commençons par rappeler les diverses méthodes de compression de données sans perte d'information, puis nous décrivons l'algorithme du contexte, et nous rapportons une évaluation comparative des performances pour divers types de fichiers. Enfin, une application au codage en sous-bande de signaux video est mentionnée.

SUMMARY

The classical approaches to data compression without loss of information, are based on a decoupling of the source model and coding. The combination most often use is statistic modelling with Huffman coding. In this case either a fixed model is assumed (which has the advantage of single pass-coding but which is not efficient for variable sources) or a model, which is adaptable to each file, but which requires a "double-pass". The single-pass incremental parsing algorithm of Ziv-Lempel achieves high compression ratios. However, this algorithm is only efficient for very large files correlated in single dimension.

To overcome these disadvantages, the "context" algorithm, which permits an adaptive markovian modelling of the source, was presented by Rissanen in 1983. With its tree modelling and a novel organization based on the minimal description length principle, this algorithm can be applied to sources correlated in several dimensions; such as images. The coding part is implemented by a multi-level multiplication free arithmetic code, which has the advantage, with respect to "prefix" codes, of perfectly recovering the entropy, whatever the statistic, and of being completely adaptive in the case of model modification.

This article is organized in three parts : first a review of the different data compression methods, without loss of compression, then a description of the "context" algorithm and a comparative performance evaluation for different types of file. Finally, an application to video sub-band coding is mentionned.

1. Introduction

Le concept de la compression de données n'est pas nouveau, il y a toujours eu un besoin d'augmenter la capacité de stockage, de reduire la durée et le coût des communications. La compression de données est un mécanisme qui transforme une séquence de données en une autre contenant la même information mais de longueur inférieure à l'originale. Pour le stockage ou la transmission de textes, de programmes ou de données cryptées, on convertit une chaîne de symboles en une suite binaire sans perte d'information, contrairement à la voix ou l'image où l'on peut tolérer une certaine dégradation. Les approches classiques de la compression de données sans perte d'information sont basées sur un découplage de la modélisation de la source et du codage. La compression n'est le fait que de la modélisation qui permet l'extraction de l'information, le codage ne représente qu'un descripteur physique de cette information, et ne pourra générer que de la redondance (nulle dans le cas d'un codage parfait).

Afin de donner un sens à cette étude comparative, il convient de classifier les différents algorithmes, nous retiendrons leur capacité à compresser un bloc de données directement sans relecture, en simple passe (par opposition aux double passe) et la faculté de la modélisation à adapter ses paramètres en fonction de la source traitée. Compte tenu des spécificités des divers algorithmes, nous pourrons alors comparer leurs performances.

Nous commencerons par une brève description du codage d'Huffman utilisé dans les méthodes double passe et simple passe non adaptative. Quand une connaissance a priori de la source n'est pas disponible et qu'une pré-analyse du bloc n'est pas possible, on est obligé d'utiliser des systèmes adaptatifs simple passe, plus souples, plus puissants mais aussi plus complexes. Dans le cadre de ces systèmes, nous présenterons au paragraphe 3, l'algorithme de Segmentation Croissante de Ziv et Lempel puis au paragraphe 4 l'algorithme du Contexte proposé par Rissanen. Une description plus détaillée de ce dernier nous permettra de nous familiariser avec les principes de la complexité stochastique et du codage arithmétique. Au paragraphe 5, un comparatif des performances des algorithmes sur différents types de données est présenté, enfin, nous mentionnerons brièvement l'application de ces techniques au codage en sous-bandes d'images ou de séquences d'images au paragraphe 6.

2. *Le codage par la méthode d'Huffman*

La modélisation statistique avec codage par la méthode d'Huffman /1/ est la combinaison la plus utilisée pour les systèmes simple passe fixes et double passe adaptatifs. Sous cette dernière forme, dans le cas d'une transmission, elle nécessite l'envoi de la table de codage ou de la statistique et l'attente de la fin du bloc de données traité pour commencer son codage, ce qui peut être pénalisant lors d'une application temps réel. Un modèle fixe simple passe du même type permet de pallier ces inconvénients. Quoique assez bien adapté à des domaines très précis comme la télécopie, ce type de modèle ne peut produire de bons résultats lors de la transmission de messages comportant des répartitions statistiques changeantes. On remarquera que dans ces divers systèmes de compression, l'algorithme d'Huffman n'a pas pour but de compresser mais de fournir le code préfix optimal en fonction de la répartition statistique évaluée. De plus, dans le cas de sources comportant un alphabet de taille réduite, le code généré peut présenter une redondance importante. Dans le cas d'un système adaptatif, toute variation de la statistique implique un recalcul complet de la table de codage, ce qui rend l'emploi de la méthode d'Huffman dans un système simple passe évolutif, particulièrement peu efficace.

3. *L'algorithme de Segmentation Croissante de Ziv-Lempel*

L'algorithme de segmentation croissante comme celui du contexte ont été développés à partir de théories sur la complexité des chaînes finies. Ziv et Lempel ont défini une mesure de la complexité /2/ qui traite uniquement les données passées sans mesure probabiliste. Leur algorithme s'adapte dynamiquement à la redondance des données compressées, il consiste à fragmenter la chaîne provenant de la source en segments de longueur croissante d'où le nom d'algorithme de Segmentation Croissante /3/. Le principe en est simple : partant avec le segment vide, chaque nouveau segment ajouté en mémoire contient un symbole de plus que le segment en mémoire le plus long pouvant reproduire les symboles suivants. Par exemple, la chaîne *s* (*ababcbcabc*) sera segmentée de la manière suivante : (*a,b,ab,c,bc,abc*) , on code alors chaque segment par le couple (*i,u*) où *i* représente l'index, en binaire, donnant la position du plus long segment correspondant aux nouveaux symboles et *u* indique le symbole ajouté pour créer le nouveau segment. Le couple représentant le segment *abc* est alors (3,*c*). La longeur de la chaîne codée est donnée par:

$$L_{ZL}(s) = \sum_{i=1}^{n(s)} \lceil \log i \rceil + n(s)$$

où $n(c)$ représente le nombre de segments obtenus et $\lceil x \rceil$ l'entier supérieur ou égal à x. On démontre que pour une chaîne infinie générée par une source stationnaire, l'algorithme est asymptotiquement optimum dans la mesure où la compression par symbole converge vers l'entropie par symbole de la source. Pratiquement, la collecte de tous les segments nécessite une mémoire très importante, en théorie le taux de compression est directement lié à la taille de la mémoire. Cependant, plus l'algorithme progresse, plus la recherche et la sauvegarde de tous les évènements passés s'avère difficile. Une description plus aisée du système va nous permettre de mieux situer les problèmes :

- segmenter la chaîne source,
- envoyer le code ou l'index qui représente le segment,
- mettre à jour le dictionnaire.

La première étape est effectuée en cherchant les évènements passés stockés dans un dictionnaire, afin de trouver le segment le plus long qui puisse correspondre aux nouveaux symboles de la chaîne source. A l'origine, cette comparaison était opérée par une "fenêtre glissante", ce qui augmentait le temps de recherche géométriquement en fonction de la progression du codage. Une structure en arbre et des tables de Hash ont permis de rendre cet accroissement linéaire. La seconde phase de l'algorithme peut présenter de nombreuses variations : si le dictionnaire est structuré en arbre, le code est un pointeur sur un noeud, dans le cas d'une table de Hash, il s'agira d'une clef pour la fonction de Hash permettant de retrouver l'adresse du segment considéré; on transmettra aussi un code pour le symbole nouveau. La mise à jour du dictionnaire est faite de manière heuristique, on peut simplement concaténer le nouveau symbole au segment trouvé pour créer un nouveau segment. Une implantation particulière de l'algorithme a été réalisé par Welch /4/ qui utilise une combinaison unique des deux dernières étapes, le code n'inclue pas le nouveau symbole, mais ce dernier sera déduit du prochain code comme étant le premier caractère du segment pointé. C'est alors que l'on pourra mettre à jour le dictionnaire pour le segment précédent. Le problème majeur de cet algorithme, de même que pour tous les modèles fonctionnant par bloc, est qu'il ne peut que traiter les corrélations existantes entre symboles d'un même bloc. Toute source corrélée sur plus d'une dimension ne peut donc être correctement traitée par cet algorithme. Pour pallier cet inconvénient, il existe une version en deux dimensions, mais, comme pour la version originale, la convergence vers un seuil optimum est très lente ce qui a tendance à limiter l'utilisation de ce type d'algorithme à des blocs de données de taille importante.

4. *L'algorithme du Contexte de Rissanen*

4.1 Introduction

L'algorithme du Contexte /8/ présente une approche plus classique du problème de la compression de données. En effet, la modélisation est Markovienne adaptative. Au lieu de partitionner une chaîne en segments significatifs de taille croissante et de collecter leur nombre d'apparitions, l'algorithme rassemble les contextes dans lesquels chaque symbole de la chaîne est apparu en association avec les nombres d'apparitions conditionnelles. Les contextes, en tant que parties de la chaîne passée, ont des tailles variables et, en général, se recouvrent. Alors que la collecte de tous les contextes possibles demanderait une place trop importante, on ne prend en compte que les plus utiles, c'est à dire ceux qui procurent le plus d'information sur la source. Cette sélection est basée sur la notion nouvelle de complexité stochastique développée depuis une dizaine d'années par Rissanen. Comme le prouvent Rissanen et Langdon /7/, une telle modélisation conditionnelle permet de tirer plus efficacement avantage de régularités statistiques. Après avoir décrit sommairement les principes de bases et le fonctionnement de l'algorithme respectivement aux paragraphes 4.2 et 4.3, nous introduirons en 4.4 les principes nouveaux de la théorie de la complexité stochastique et ceux du codage arithmétique /5/ en 4.5. L'application de ces notions permet de tirer parti de manière optimale de la modélisation définie par l'algorithme.

4.2 Principe de la creation des contextes

Un contexte est défini par une fonction récursive $f: A \rightarrow N$, avec A: l'ensemble de toutes les chaînes finies et N: l'ensemble des nombres naturels. Le contexte z(t) du symbole x(t), suivant immédiatement la chaîne s = x(1)....x(t-1) peut être défini comme la classe d'équivalence des chaînes s' = x'(1)...x'(t-1) telle que f(s') = f(s). Plus pratiquement, le contexte de x(t) est une fonction de la chaîne passée. Le fait que le contexte soit une classe d'équivalence signifie simplement que toutes les chaînes passées distinctes ne définissent pas forcément des contextes distincts (plusieurs chaînes pouvant appartenir à la même classe d'équivalence et donc définir le même contexte). On définit une relation d'ordre pour le choix des symboles passés devant constituer le contexte du symbole x(t). En considérant en premier lieu les symboles passés que l'on juge comme ayant la plus grande influence sur la valeur de la variable x(t).

4.3 Mémorisation des contextes

L'idée est de stocker pour chaque symbole de l'alphabet de la source les contextes dans lesquels ils sont apparus dans un arbre (figure 1). Soit T_a la taille de l'alphabet, on devrait avoir T_a arbres. En réalité, on ne construit qu'un seul arbre, chaque noeud comprenant alors T_a compteurs. On notera c(u,z) le compteur d'occurence du symbole u dans le contexte z. La création et la mise à jour de l'arbre se fera de la manière suivante :

- Initialiser les T_a compteurs de la racine, c'est à dire du premier noeud, à 0.

En procédant récursivement, aprés l'apparition de chaque nouveau symbole x(t), incrémenter le compteur c(x(t),racine) d'une unité. En partant de la racine, prendre les branches menant aux noeuds d'indice z_1,z_2 ... . Pour chaque noeud z visité, incrémenter d'une unité le compteur c(x(t),z). Continuer jusqu'à ne plus trouver de noeud z existant.

- Arrivé a un noeud terminal, si le compteur incrémenté était auparavant égal à un, créer un nouveau noeud z en initialisant à 1 le compteur c(x(t),z) et à 0 les (T_a-1) autres compteurs. On appelera T(t) la mise à jour de l'arbre pour le symbole x(t).

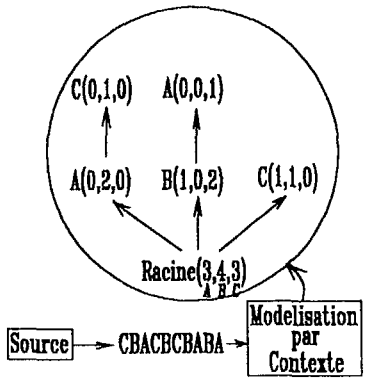

figure 1. Exemple de création de l'arbre des Contextes

Il est cependant possible d'appliquer d'autres règles pour le développement de l'arbre. Afin de le rendre moins redondant, on peut par exemple attribuer un coût à chaque noeud en fonction de son entropie /8/, ou lier le développement de l'arbre et le choix du contexte de codage /11/. Deux problèmes se posent : la mise à jour et le developpement de l'arbre d'une part, le choix du noeud (du contexte) le plus approprié pour l'estimation de la probabilité du symbole x(t) et son codage d'autre part. Ce dernier point est capital car il conditionne l'efficacité de l'algorithme. Lorsqu'on suit un chemin dans l'arbre pour sa mise à jour, chaque noeud rencontré représente une distribution de probabilités dans un contexte donné. Dans le cas d'une transmission, le décodeur ne connaissant pas x(t), il faut que le choix du noeud de codage soit indépendant de x(t) et que l'estimation de la probabilité de x(t) soit effectuée avant la mise à jour de l'arbre. Il faut donc une règle qui optimise ce choix. Le modèle ainsi généré peut être considéré comme une approximation de la complexité stochastique de la source traitée. La gestion du modèle étant alors basée sur le principe de la Longueur de Description Minimale /9/.

4.4 Introduction à la Complexité Stochastique

L'information relative à une chaîne x produite par une source donnée, suivant une fonction de probabilité P, est donnée par -log P(x). Alors que pour des codes universels, une mesure acceptable de leurs performances est fournie par la longueur moyenne dans le cas le plus défavorable pour une classe de sources, l'idée de la Complexité cherche à fournir la longueur moyenne minimale pour chaque source. Dans le cas qui nous interesse, c'est à dire la classe de sources pouvant être représentées par une machine à états finis et plus particulièrement, pour les sources Markoviennes, Rissanen à démontré /9/, que l'application de cette notion aboutissait, asymptotiquement, à l'obtension de la longueur moyenne minimale pour le code généré. La Complexité Stochastique d'une chaîne s relativement à une classe de source pouvant être considérée comme des machines à états finis (ce qui est le cas pour les sources Markoviennes) est définie par:

$$I(x) = \min_i \{ -\log P(s|f_i) + D(i) \}$$

Où f_1,f_2, ... représentent l'ensemble des fonctions ordonnées suivant leur nombre d'états et D(i) le coût de description du modèle considéré.

4.5 Application du principe de la LDM pour l'optimisation du modèle

Pratiquement, on ne peut essayer tous les modèles de tous ordres. On travaillera donc sur un seul que l'on essayera de faire évoluer de manière optimale avec un seul estimateur. La recherche du modèle simplifié nécessite pour chaque noeud une optimisation locale afin que le modèle déterminé à partir de l'arbre T(t) constitue pour le codage de x(t+1) une bonne approximation du modèle optimum. Cette optimisation fait appel au principe de la Longueur de Description Minimale (LDM). Connaissant le chemin dans l'arbre représenté par le contexte z_1,z_2, ..., on choisit comme noeud z de codage le noeud qui aurait généré la longueur minimale pour le codage de la chaîne s. Soit L_z la longueur de code générée par un noeud z et définie par :

$$L_z = \sum_{i=1}^{t} -\log \hat{P}(x_i|z)$$

On choisit alors le premier noeud tel que :

$$L_z \leq \sum_{i=1}^{T_a} L_{zi}$$

en désignant par zi le noeud fils du noeud z, d'indice i. Compte tenu d'un certain contexte, on choisit l'ordre de Markov optimal. Grâce à l'application de ce principe pour l'optimisation locale du modèle, on obtient pour la source une bonne approximation de sa complexité stochastique et donc une longueur de code généré voisine du minimum.

4.6 Implantation de l'algorithme

Pratiquement, si l'on désire implanter cet algorithme, on doit minimiser l'occupation mémoire et le temps de calcul par symbole. Grâce à un système de pointeur, un noeud ne comprendra que les compteurs non nuls de même que l'arbre ne contiendra que les noeuds représentant un contexte apparu au moins une fois dans la chaîne, on peut, au fur et à mesure, rajouter des compteurs aux divers noeuds de même que des noeuds nouveaux. Ce système de pointeurs simplifie la recherche des compteurs nécessaires à l'estimation de la probabilité ou au choix du noeud de codage et diminue le temps de calcul. Une fois la probabilité du nouveau symbole estimé, on la code grâce à un code arithmétique multi-niveaux si l'alphabet de la source n'est pas binaire /6/. Ce type de codage ne comporte pas de multiplication ce qui le rend très rapide. La longueur moyenne par symbole est très proche de l'entropie théorique (entre 2 et 3 % de différence) quelque soit la taille de l'alphabet. Il est enfin parfaitement adaptatif ce qui est nécessaire puisque chaque symbole codé fait appel à une distribution statistique différente en fonction du noeud de codage.

4.7 Introduction aux codes arithmétiques

Afin de simplifier cette présentation, nous travaillerons avec un alphabet binaire, alphabet normalement incompressible par la méthode d'Huffman sans extension d'alphabet. Notre exemple aura pour but de compresser la chaîne binaire 11010 ayant auparavant estimé les probabilités des symboles 0 et 1. L'algorithme de codage commence par diviser l'intervalle probabilité [0,1] proportionnellement aux probabilités des symboles 0 et 1, (sans perte de généralité, on supposera que l'intervalle correspondant au "0" sera situé en bas). Puisque le premier élément de la séquence d'entrée est un "1", on selectionne le sous-intervalle supérieur pour de futures subdivisions. Ce sous-intervalle est alors subdivisé proportionnellement aux probabilités du 0 et du 1, et le nouveau sous-intervalle correpondant au deuxième élément de la séquence d'entrée (un 1 de nouveau), est choisi pour les prochaines subdivisions. Ce processus est répété jusqu'à ce que l'on arrive à l'intervalle A_s, qui est l'unique représentation de la séquence d'entrée. L'unique manière d'arriver à l'intervalle A_s est obtenu par la séquence 11010. En d'autres termes, une autre représentation possible de cette séquence est obtenue par un nombre contenu dans l'intervalle A_s. De plus, ce nombre n'a besoin que d'un nombre de digits suffisant pour identifier son appartenance à l'intervalle A_s. Du point de vue de la compression, on montre /10/ que le nombre de digits nécessaire approche l'entropie de la séquence d'entrée.

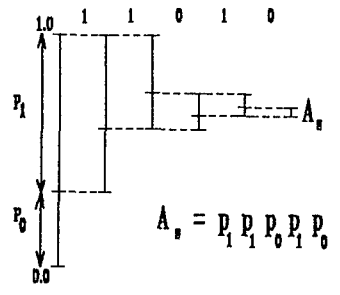

figure 2. Codage de la séquence 11010

5. *Comparatif des performances*

Tous les tests ont été réalisés en simulation. Les algorithmes d'Huffman et de Ziv-Lempel etaient déjà disponibles sur le système sous forme de produit programme appliqué à la compression de données. L'algorithme du Contexte à été testé en temps réel avec des tailles mémoire allant de 2 à 32 Kmots. Quelque soit le fichier traité, c'est ce dernier algorithme qui a fourni les meilleurs résultats. Ces travaux ont été menés dans le cadre d'études sur le traitement d'images fixes et vidéo afin de les intégrer dans les algorithmes de compression, essentiellement basés, comme on le verra au paragraphe suivant, sur les techniques de codage en sous-bandes. Des tests effectués avec différentes techniques de compression d'images, sans perte d'information, associées au Contexte, conduisent à des taux de compression, de l'ordre de 4.

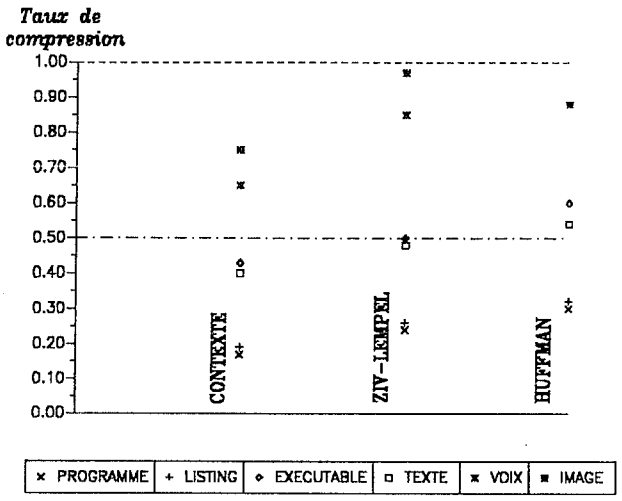

figure 3. Comparatif des performances pour divers types de fichiers

6. *Codage en sous-bandes d'images fixes et vidéo*

De même que pour des données de type texte ayant un alphabet fini, les techniques de codage entropique peuvent être utilisées pour des signaux quantifiés comme dans le cas des systèmes de codage de la parole ou de l'image. Les choix du modèle de représentation du signal à traiter et des quantificateurs sont responsables de la perte d'information d'une part et du taux de compression obtenu d'autre part. Le codage entropique alors appliqué aux signaux quantifiés permet une compression supplémentaire, sans perte, de la quantité d'information à transmettre.

D. Le Gall et A. Tabatabai /12/ ont appliqué le codage arithmétique au codage en sous-bandes d'images fixes. De même que G. Karlsson et M. Vetterli /13/, nous envisageons le codage entropique dans le cas d'une décomposition en sous-bandes de séquences d'images vidéo. Une réduction du nombre de niveaux à coder en sortie du quantificateur est assimilable à un alphabet de taille réduite pour lequel un codeur entropique s'avère très efficace. Les travaux préliminaires que nous avons conduit montrent que le codage en sous-bandes appliqué à l'image, utilisé en conjonction avec un codeur arithmétique conduit à une réduction sans perte d'information d'un facteur 4 environ (2 bits/pixel), il est possible d'atteindre un facteur 16 (0.5 bits/pixel) tout en conservant une bonne qualité au plan perceptuel. Dans le domaine de la vidéo, nous nous intéressons au concept de codage en sous-bandes multi-débit. Il permet d'utiliser le principe de division du spectrale du codage en sous-bandes pour définir une gamme de codeurs proposant des niveaux de qualités différents, à mesure que l'on prend en compte un plus grand nombre de sous-bandes à coder ou que l'on code plus finement chaque sous-bande, et correspondant à des taux de compression différents.

7. *Conclusion*

Cette étude nous a permis de faire le point sur les divers systèmes de compression de données sans perte d'information, il faut cependant remarquer que le Contexte est bien plus qu'un système de compression, il permet dans sa version intégrale de donner une limite à la compression d'une source, de plus, l'arbre qu'il génère est un élément précieux d'apréciation de la source codée, et permet de la caractériser facilement. Dans tous les cas, un système de compression de données permet de réduire, de manière importante la redondance de la source et d'optimiser l'utilisation de l'espace de stockage ou du temps de transmission.

7. *Bibliographie*

/1/ R.Gallager "Variations on a theme by Huffman".
IEEE, Trans. Inform. theory, Vol IT24, No 6, Nov. 1978

/2/ J.Ziv, A.Lempel "On the complexity of finite sequences".
IEEE, Trans. Inform. theory, Vol IT22, No 1, Jan. 1976

/3/ J.Ziv, A.Lempel "Compression of individual sequences via variable-rate coding".
IEEE, Trans. Inform. theory, Vol IT24, No 5, Sept. 1978

/4/ T.Welch "A technique for high performance data compression".
IEEE, Comput. Mag., June 1984

/5/ J.Rissanen, G.G.Langdon "Arithmetic coding".
IBM Journal of Research and development, vol 23, No 2, Mars 1979

/6/ J.Rissanen, K.Mohiuddin "A multiplication-free arithmetic code".
IEEE, Trans. Inform. theory, à paraître

/7/ J.Rissanen, G.G.Langdon "Universal Modeling and Coding".
IEEE, Trans. Inform. theory, Vol IT27, No 1, Jan. 1981

/8/ J.Rissanen "A universal data compression system"
IEEE, Trans. Inform. theory, Vol IT29, No 5, Sept. 1983

/9/ J.Rissanen "Complexity of strings in the class of Markov sources".
IEEE, Trans. Inform. theory, Vol IT32, No 4, July 1986

/10/ R.Pasco "Source coding algorithms for fast data compression".
Ph. D. Thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, May 1976.

/11/ G.Furlan "Compression de données et d'images".
Thèse Ph. D. Department of Advanced Technology in Systems Centre d'Etudes et Recherches IBM La Gaude, à paraître

/12/ D. Le Gall, A. Tabatabai "Sub-band coding of digital images using symetric short kernel filters and arithmetic coding techniques".
Proceedings ICASSP, New York, April 11-14, 1988.

/13/ G. Karlsson, M. Vetterli "Three dimensional sub-band coding of video".
Proceedings ICASSP, New York, April 11-14, 1988.