N° d'ordre : ……..

UNIVERSITE DE M'SILA
FACULTE DE TECHNOLOGIE
DEPARTEMENT D'ELECTRONIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de Magistère

Spécialité : Electronique

Option : Communication

Par
GHAZALI FATEH

Sujet

LA COMPRESSION DE DONNEES EN UTILISANT LA TRANSFORMEE BURROWS-WHEELER (BWT)

Soutenu publiquement le 11 avril 2011 devant le jury composé de :

Mr. Djamel CHIKOUCHE, Professeur, Université de M'SILA, **Président**
Mr. Redha BENZID, Maître de conférences A, Université de BATNA, **Rapporteur**
Mr. Mohamed BOUAMAR, Maître de conférences A, Université de M'SILA**, Examinateur**
Mr. Khireddine CHAFAA, Maître de conférences A, Université de BATNA, **Examinateur**
Mr. Djamel SAIGAA, Maître de conférences A, Université de M'SILA, **Examinateur**

Promotion : 2010/2011

REMERCIEMENT ET GRATITUDE

Je tiens à adresser mes remerciement les plus profonds à mon rapporteur monsieur **Benzid Redha**, maître de conférences à l'université de Batna qui m'a guidé avec grande patience tout au long de l'élaboration de ce travail et pour ces aides précieuses qui ont judicieusement éclairé mon chemin vers l'aboutissement et la concrétisation de mon mémoire.

Je remercie vivement les membres de jury qui ont accepté de juger ce modeste travail :

Monsieur **Djamel CHIKOUCHE**, Professeur, Université de M'SILA, qui m'à honoré par sa présence en qualité de Président de jury.

Monsieur **Mohamed BOUAMAR**, Maître de conférences A, Université de M'SILA, pour l'honneur qu'il me fait en acceptant de participer à ce jury.

Monsieur **Khireddine CHAFAA**, Maître de conférences A, Université de BATNA, pour l'honneur qu'il me fait également en acceptant de participer à ce jury.

Monsieur **Djamel SAIGAA**, Maître de conférences A, Université de M'SILA, qui m'a honoré avec son acceptation de juger le travail effectué.

Enfin, rien ne se serait passé sans la présence de mon entourage, ma femme, ma mère mes enfants et mes aimés. Un grand merci à tous.

Table de matière

Chapitre III : Transformée de Burrows Wheeler

Chapitre IV : Applications

Liste des figures :

Liste des tableaux:

Introduction générale

La compression de données est un vaste sujet qui a fait l'objet de nombreux ouvrages et articles [9,10,13,26,30]. Elle donne lieu aujourd'hui à de nombreuses recherches en raison des enjeux économiques. Elle est utilisée majoritairement dans les applications informatiques et elle peut être considérée comme l'une des conditions d'existence du multimédia.

Les algorithmes de compression sont utilisés pour réduire toute redondance dans la représentation de données, permettant ainsi de diminuer l'espace de stockage nécessaire. Par conséquent, la compression offre une approche attrayante pour réduire les coûts de communication en employant efficacement la largeur de bande d'un canal de transmission et/ou une utilisation optimale d'un support de sauvegarde.

Pendant les dernières années il y a eu une explosion dans la quantité de données numériques transmises par l'intermédiaire de l'Internet, incluant le texte, les images, la vidéo…..etc. [9]

Dans ce contexte, ce mémoire est consacré à dresser le problème de la compression de données (textes, image niveaux de gris), en utilisant la transformée de Burrows Wheeler (BWT). Cette dernière est utilisée comme une phase de prétraitement dans la chaîne de compression appliquée à une donnée afin de la rendre plus compressible. Elle est souvent employée en association avec les algorithmes - Move-to-Front (MTF), RLE, Huffman ou le codage arithmétique.

En exploitant sa capacité de tri de données, nous essayons de mettre le point sur l'utilité du réarrangement des données initiales, par l'utilisation de la BWT, dans un schéma de compression, que ce soit pour texte ou image.

Ce manuscrit est divisé en quatre chapitres.

Le premier chapitre fournit une introduction à la théorie d'information et les différentes techniques employées dans le codage de données.

Le deuxième chapitre est consacré à la compression sans et avec perte de données et les différents algorithmes employés.

Le troisième chapitre aborde la transformée de Burrows Wheeler et l'algorithme Move-To-Front (MTF).

Le quatrième chapitre est dédié à la simulation et à l'implémentation de la méthode sur deux types différents de données (texte et image à niveaux de gris).

Et enfin, une conclusion résumant le travail effectué ainsi que la proposition de quelques travaux futurs.

Introduction à la théorie de l'information

I-1- Historique

Beaucoup de théories de la communication moderne provient des travaux de Shannon (1948), de Wiener (l949), et de Kotel'nikov (1947).Tous ces hommes ont identifié clairement le rôle fondamental du bruit en limitant l'exécution des systèmes de communication et également de modeler le signal et le bruit en tant que processus aléatoires. Wiener s'intéressait à trouver le meilleur filtre linéaire pour séparer le signal du bruit additif et son travail a eu une influence importante sur les recherches dans la théorie de modulation. Le travail de Kolmogorov (l941) sur la prévision en l'absence du bruit a eu un impact important sur la théorie de commande. De même, Kotel'nikov il s'intéressait à la détection et l'évaluation des signaux au récepteur.

Le travail de Shannon a eu une saveur beaucoup plus numérique que les autres, et a eu plus d'importance par sa concentration conjointement sur l'encodeur et le décodeur. En raison de cette emphase commune et en l'absence des restrictions aux types particuliers de structures de récepteur, la théorie de Shannon fournit le cadre conceptuel le plus général connu pour étudier une communication efficace et fiable. [2]

I-2- Introduction

Les concepts et les principes généraux de la théorie de l'information ont été développés par Claude E. Shannon dans les années 40, tandis qu'il travaillait aux Laboratoires Bell, ingénieur en génie électrique qui formalise mathématiquement la nature statistique de l'information dans les signaux des lignes téléphoniques. Pour ce faire, il développa le concept général d'entropie et d'information. En conséquence, nous allons étudier cette entropie (qui se trouve être la même, à une constante multiplicative prés, que celle de Boltzmann) et ses liens avec la compression de données. Ainsi, l'apparition de l'information et de la communication numérique ont été à la base de la reconsidération de

la théorie de l'information sous une nouvelle vision. En raison de la théorie de l'information et des résultats obtenus de la théorie de codage il est possible de quantifier l'information, ce qui permet de la coder et la transmettre efficacement. [1]

La théorie de l'information répond à deux questions fondamentales dans la théorie de la communication : Quelle est la limite de la compression de données finale (l'entropie *H) ?* Et quel est le taux final de transmission (la capacité de canal *C) ?* Pour cette raison certains considèrent la théorie de l'information comme un sous-ensemble de la théorie de la communication. . En effet, elle a des contributions fondamentales dans plusieurs domaines tels que : La physique statistique (thermodynamique), Informatique (complexité de Kolmogorov ou complexité algorithmique), Inférence statistique (le rasoir d'Occam) et aux probabilités et aux statistiques (taux d'erreur pour l'essai optimal et l'évaluation d'hypothèse). [2]

I-3- Mesure de l'information

Le développement de l'idée de l'entropie des variables aléatoires et des processus est lancé la première fois par Claude Shannon et celui qui a fourni les commencements de la théorie de l'information et les techniques modernes de la théorie ergodique. Nous présentons par la suite les diverses notions de l'entropie pour des variables aléatoires. [5]

Dans cette section nous donnons une mesure de la quantité d'information adaptée à la description statistique des sources et des canaux. Les énoncés qui en résultent faisant largement appel aux probabilités discrètes.

I-3-1- Quantité d'information

Au plan le plus général, la question de la mesure de l'information se pose ainsi : On considère un système S pouvant revêtir un certain nombre M d'états ou de configurations possibles et par conséquent seulement probables aux yeux d'un observateur O. La

constatation par O que l'état M_i s'est réalisé constitue l'acquisition d'une information relative à S et à ses configurations potentielles [4]. On écrit

$$I=f(M) \qquad \text{(I-1)}$$

Le choix de la fonction f est dicté par le bon sens.

Le nombre d'états possibles de deux systèmes indépendants S_1, S_2 est le produit M_1 M_2 du nombre d'état de chacun d'eux. Il est raisonnable d'écrire que l'information correspondante est I_1+I_2, d'où l'identification de f à la fonction logarithme. On écrit maintenant :

$$I = K \log M \qquad \text{(I-2)}$$

Cela revient à mesurer l'information par la richesse des potentialités des systèmes qui la portent ou par leur diversité.

De plus, S étant capable de M configurations équiprobables, la probabilité a priori de chacune d'elles, vue de O est : $p = \frac{1}{M}$ ou $I = -K \log p$ (I-3)

Cela revient ainsi à mesurer l'information par la rareté d'une configuration d'un système. Cela est conforme à l'idée commune, « on en apprend d'autant plus qu'on ne s'y attendait pas ».On convient d'adopter ce point de vue pour définition générale de l'information :

La quantité d'information I attachée a une configuration i quelconque d'un système dont la probabilité a priori est P_i pour un observateur externe O, s'écrit :

$$I = -K \log p_i \qquad \text{(I-4)}$$

L'observateur qui a pris connaissance de cette configuration a reçu, a posteriori, la quantité d'information I.

I-3-2- Unité d'information

Le choix d'unité d'information c'est-à-dire de la constante K et de la base des logarithmes est évidement libre.

On s'adresse au cas le plus simple, celui de deux états possible, indépendants et a priori équiprobables. On décide que leur connaissance a apporté une unité d'information.

On voit que l'égalité $I = 1 = -K \log 2$ est satisfaite pour K=1

Dans ces conditions, la formule de quantité d'information précédentes s'écrite :

$$I = -\log_2 p_i \qquad \text{(I-5)}$$

L'unité d'information s'appelle le **Shannon (Sh)**

Le Shannon est la quantité d'information reçue par un observateur, après lever de doute entre deux éventualités, a priori équiprobables, des états d'un système.

I-4- Entropie et Information mutuelle

Le concept d'information est trop large pour être capturé complètement par une simple définition. Cependant, pour n'importe quelle distribution de probabilité, nous définissons une quantité appelée *l'entropie*. Cette notion est prolongée pour définir *l'information mutuelle,* qui est une mesure de la quantité de l'information d'une variable aléatoire contient des liaisons avec autres variables aléatoires. Donc, l'information mutuelle est un cas spécial d'une quantité plus générale appelée *l'entropie relative,* qui est une mesure de la distance entre deux distributions de probabilité. Toutes ces quantités sont étroitement reliées et partagent un certain nombre de propriétés simples. Nous décrivons certaines de ces propriétés en ce chapitre.

I-4-1- Information mutuelle

Dans l'espace probabilisé joint XY , ou $X = \{a_1, \ldots, a_K\}$ et $Y = \{b_1, \ldots, b_J\}$, l'information mutuelle peut être considérée comme une variable aléatoire réelle.

L'information mutuelle moyenne de X et Y dans l'espace probabilisé joint XY est définie par :

$$I(X,Y)=\sum_{k=1}^{K}\sum_{j=1}^{J}P(a_k,b_j)I(a_k,b_j)=\sum_{k=1}^{K}\sum_{j=1}^{J}P(a_k,b_j)\log_2\frac{P(a_k,b_j)}{P(a_k)P(b_j)} \quad \text{(I-6)}$$

On peut également définir la moyenne de l'information propre d'un espace probabilisé $X = \{a_1, \ldots, a_K\}$, cette moyenne porte le nom d'entropie.[3]

I-4-2- Entropie (information moyenne)

L'entropie désigne la quantité d'information que peut représenter ou contenir une source d'information. Ici, une source d'information est généralement un fichier informatique quelconque (texte, image, vidéo, son……..). Plus une source d'information est redondante, moins elle contient d'information au sens de Shannon.

On suppose un fichier ou un ensemble de données comme étant une source d'information. Cette dernière est considérée comme un ensemble de séquences (caractères) issues d'un alphabet caractéristique à cette source. Chaque caractère de l'alphabet est délivré avec une fréquence P_i, la quantité d'information moyenne apportée par cette source (ex : image) est identifiée à L'entropie [2]

L'entropie est aussi la plus petite quantité moyenne de bits nécessaire pour communiquer la vraie valeur d'une variable aléatoire. En d'autres mots, l'entropie est la plus petite quantité moyenne de bits capable de représenter l'information tirée de la variable aléatoire. Il s'agit d'une limite mathématique fondamentale pour la compression de données sans perte**.** [7]

Nous présenterons d'abord le concept de l'entropie, qui est une mesure d'incertitude d'une variable aléatoire. Soit X une variable aléatoire discrète dans l'alphabet *A* et *p(x)*, la fonction de probabilité $p(x) = \Pr\{X = x\}$, $x \in A$, *l*'entropie d'un espace probabilisé X est définie par :

$$H(X)=\sum_{k=1}^{K}P(a_k)I(a_k)=\sum_{k=1}^{K}-P(a_k)\log_2 P(a_k) \quad \text{(I-7)}$$

On aborde maintenant le cas général, celui où les états d'une source d'information sont indépendants, porteur d'information, ne sont plus équiprobables. On suppose que le récepteur R reçoit une succession de N messages, l'état i du système, de probabilité a priori p_i étant codé (représenté) par le message n_i

Donc :
$$I_N = -\sum_1^n n_i \log_2 p_i \qquad \text{(I-8)}$$

Soit, en moyenne par état ou message :

$$\frac{I_N}{N} = -\sum \frac{n_i}{N} \log_2 p_i \qquad \text{(I-9)}$$

Cette quantité tend vers $-\sum p_i \log_2 p_i$ quand $N \to \infty$.

On l'appelle information moyenne par état ou par message, mot ou lettre, ou encore l'entropie H :

$$H = -\sum p_i \log_2 p_i \qquad \text{(I-10)}$$

Cela revient simplement à former la moyenne en probabilité des quantités d'information. L'entropie est identifiée à l'espérance mathématique. Dans un ensemble de n messages différents possibles, les informations qu'ils portent sont pesées au poids de leur probabilités.[4]

I-4-3- Entropie et la compression

Nous pouvons citer les remarques suivantes :

♦ Le codage ASCII utilisé dans les PC (personnel computer) est un codage non optimal (au sens de la compression), du fait qu'il suppose qu'une source d'information (fichier) est constituée d'un alphabet de 256 caractères ce qui n'est pas vrai dans la plus part des cas.

♦ Dans les sources qui ont une distribution uniforme (l'entropie égale l'entropie d'ordre 0), le taux de compression d'un compresseur efficace sans perte (par exemple WINRAR) est pratiquement égale un 1 (il n'y a pas de compression), parce que l'entropie d'ordre 0

est la mesure indiquant qu'il n'existe pas une méthode de compression sans perte qui peut descendre au dessous. Dans le meilleur cas une méthode de codage converge vers l'entropie. [8]

I-5- Systèmes de communication

La théorie des communications s'intéresse aux moyens de transmettre une information depuis une source jusqu'à un utilisateur (Figure I.1). La nature d'une source d'information peut-être très variée.

Il peut s'agir par exemple d'une voix, d'un signal électromagnétique ou d'une séquence de symboles binaires. Le canal peut être une ligne téléphonique, une liaison radio ou encore un support magnétique ou optique : bande magnétique ou disque compact. Le canal sera généralement perturbé par un bruit qui dépendra de l'environnement et de la nature du canal : perturbations électriques, rayures, etc… Le codeur représente l'ensemble des opérations effectuées sur la sortie de la source avant la transmission. Ces opérations peuvent être, par exemple, la modulation, la compression ou encore l'ajout d'une redondance pour combattre les effets du bruit. Elles ont pour but de rendre la sortie de la source compatible avec le canal.

Enfin, le décodeur devra être capable, à partir de la sortie du canal de restituer de façon acceptable l'information fournie par la source.

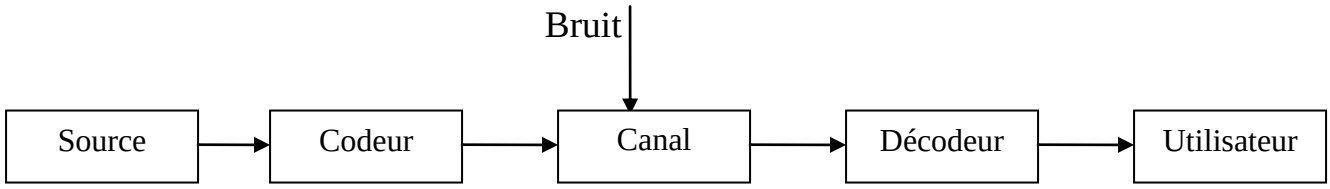


Figure I-1 : Schéma d'un système de communication numérique [1]

Lors des années 40, où Shannon a développé la théorie de l'information celle-ci s'est faite de plus en plus précise et est devenue aujourd'hui incontournable dans la conception de tout système de communication, au sens le plus large de ce terme. [3]

Dans cette section, nous étudierons certains de ces modèles mathématiques, qui, bien que considérablement plus simple que les sources et les canaux physiques, permettent de donner une bonne intuition de leur comportement.

Dans le but de simplifier l'étude des systèmes de communication, nous étudierons séparément les modèles de sources et les modèles de canaux. Ceci peut se schématiser en séparant le codeur et le décodeur de la Figure I.2 en deux parties. Le but du codeur de source est de représenter la sortie de la source, ou information, en une séquence binaire, et cela de la façon la plus économique possible. Le but du codeur de canal et de son décodeur est de reproduire le plus fidèlement possible cette séquence binaire malgré le passage à travers le canal bruité.

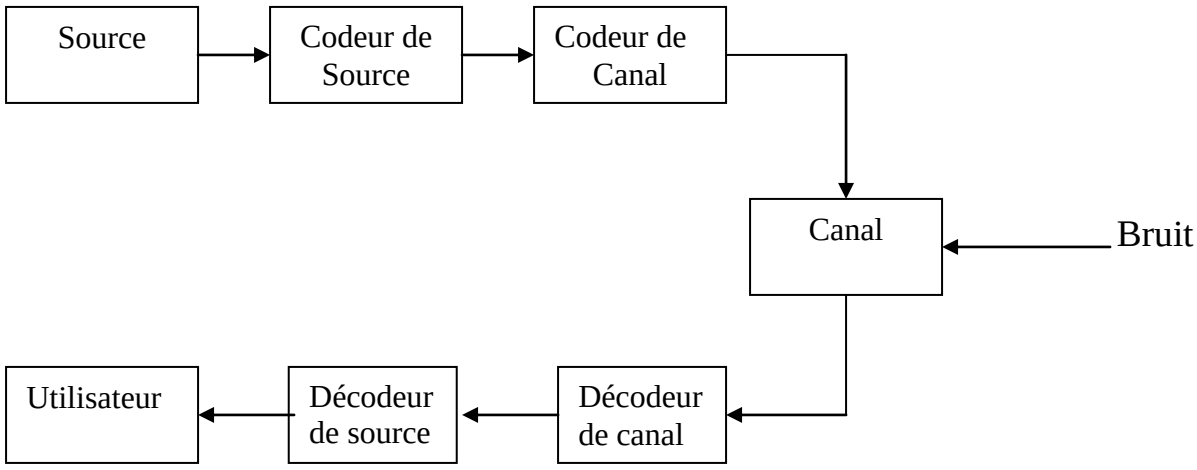


Figure I-2 : Système de communication avec codeurs de source et de canal séparés.

Cette séparation rendra notre étude commode, car nous pourrons traiter indépendamment la source et le canal. De plus cette séparation n'implique aucune limitation sur les performances du système. Nous poursuivrons donc ce chapitre d'introduction par une brève description des différentes classes de modèles de sources et de canaux. [3]

I-5-1 Sources et codage de source

Parmi les classes possibles de modèles de sources, nous nous intéresserons plus particulièrement aux *sources discrètes sans mémoire*. La sortie d'une telle source est une séquence de lettres tirées dans un alphabet fini { a_1, a_2,……a_K }. Chaque lettre de la séquence est statistiquement indépendante des autre et obtenue d'après une loi de probabilité fixe $P(a_1)$, . . . , $P(a_K)$. Pour toute lettre ak, $1 \leq k \leq K$, de la source, $P(a_k)$ est la probabilité pour que cette lettre soit produite, nous aurons donc :

$$\sum_{k=1}^{k=K} P(a_k) = 1 \qquad \text{(I-11)}$$

Il peut sembler étonnant de modéliser une source d'information à l'aide d'un processus aléatoire. L'exemple suivant nous montre toutefois que l'utilisation des probabilités est indispensable pour coder une information de façon compacte. [2]

Exemple:

Soit une source d'information qui fournit l'une des quatre lettres a_1, a_2, a_3 et a_4. Supposons que le codage transforme cette information en symboles binaires. Dans le Tableau I.1, nous donnons deux codages différents de cette source.

Tableau. I-1: Deux codages d'un alphabet de quatre lettres.[3]

Méthode 1	Méthode 2
$a_1 \longrightarrow$ 00	$a_1 \longrightarrow$ 0
$a_2 \longrightarrow$ 01	$a_2 \longrightarrow$ 10
$a_3 \longrightarrow$ 10	$a_3 \longrightarrow$ 110
$a_4 \longrightarrow$ 11	$a_4 \longrightarrow$ 111

Dans la première méthode, deux symboles binaires sont générés pour chaque lettre émise, alors que dans la seconde le nombre de symboles est variable. Si les quatre lettres sont

équiprobables, alors la première méthode est la meilleure : 2 symboles par lettre en moyenne au lieu de 2,25. En revanche si l'on a :

$P(a_1) = 1/2$, $P(a_2) = 1/4$, $P(a_3) = P(a_4) = 1/8$

Alors la méthode 1 nécessite toujours 2 symboles binaires par lettre en moyenne alors que la méthode 2 ne nécessite que 1,75. Elle est dans ce cas la plus économique. Il est donc important pour coder correctement une source de connaître son comportement [6]

I-5-2- Canaux et codage de canal

Pour modéliser correctement un canal de transmission il est nécessaire de spécifier l'ensemble des entrées et l'ensemble des sorties possibles. Le cas le plus simple est celui du *canal discret sans mémoire* : l'entrée est une lettre d'un alphabet fini $\{a_1, \ldots, a_K\}$, et la sortie est une lettre d'un autre alphabet fini, $\{b_1, \ldots, b_J\}$. Ces lettres sont émises en séquence, et pour que le canal soit sans mémoire, il faut que chaque lettre de la séquence reçue ne dépende statistiquement que de la lettre émise de même position. Ainsi un canal discret sans mémoire est entièrement décrit par la donnée des probabilités conditionnelles $P(b_j \mid a_k)$, pour tout a_k dans l'alphabet d'entrée et tout b_j dans l'alphabet de sortie.

Par exemple le canal binaire symétrique, représenté dans la Figure I.3, est un canal discret sans mémoire, dont l'alphabet d'entrée et l'alphabet de sortie sont égaux tous deux à $\{0, 1\}$. La probabilité pour qu'un symbole soit inchangé est $1 - \varepsilon$, et la probabilité pour qu'il soit transformé en son opposé est ε.

On peut également considérer des canaux discrets à mémoire dans lesquels chaque lettre de la séquence de sortie peut dépendre de plusieurs lettres de la séquence d'entrée.

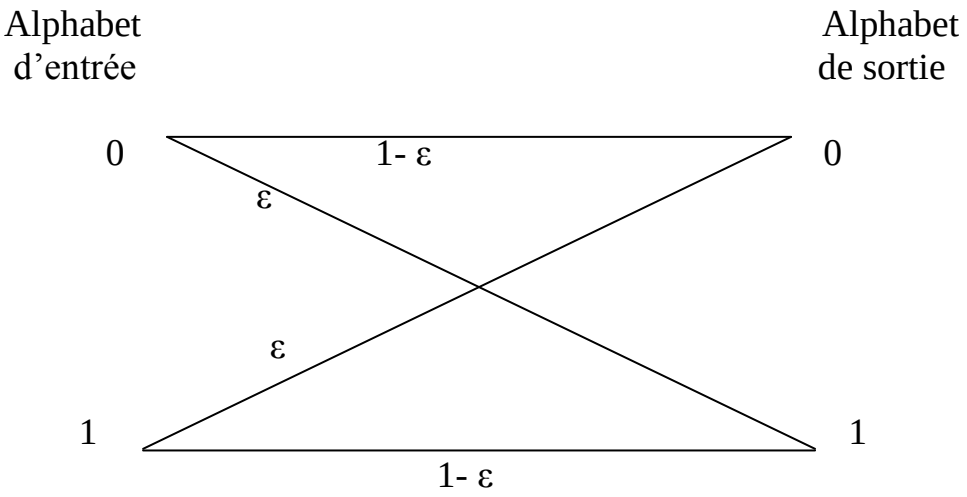


Figure I-3 – Canal binaire symétrique.[3]

I-6- Conclusion

Dans ce chapitre, nous avons défini l'information, l'information mutuelle et le développement de certain nombre de propriétés d'information qui seront utiles en chapitres suivants. Ces propriétés pour la plupart ce que nous compterions d'une définition mathématique raisonnable d'information, mais la vraie justification pour les définitions viendront dans les chapitres suivants. Afin d'aborder la compression d'une manière rigoureuse, nous avons étudié certains résultats fondamentaux de théorie de l'information utilisés en compression. Un développement plus détaillé, ainsi que les preuves de la plupart des résultats présentés se trouvent dans [3]. Cette théorie, introduite par Claude Shannon en 1948, établit les limites atteignables pour la communication et la compression de données numériques. En particulier, il est prouvé que les réalisations d'un processus aléatoire discret dont on connaît la statistique (une source) ne peuvent être représentées sans erreur en un nombre moyen de bits inférieur à une constante appelée l'entropie de la source (théorème de codage source),

Compression de données

II-1- Introduction

La compression de données (data compression) est considérée comme l'un des axes de recherche les plus actifs appartenant à la théorie de l'information.

Elle s'impose dans plusieurs applications scientifiques, industrielles,…..

C'est grâce aux techniques élaborées que l'Internet s'est infiltré de plus en plus dans nos besoins journaliers de données numériques (compression de vidéo, de sons, de données médicales, géophysiques, ….)

L'autre exemple confirmé de son application est le réseau de la téléphonie mobile exigeant un flux d'échange d'informations multimédia énorme. En conséquence, l'emploi de la compression, réduisant ainsi le temps de transmission, a rendu le téléphone mobile un véritable moyen efficace d'échange de données numériques.

Dans ce chapitre nous allons introduire les notions nécessaires pour avoir un aperçu sur les différents algorithmes de compression en montrant, le plus possibles, leurs aspects techniques.

II-2- Définition de la compression de données

La compression de données est toute technique permettant de réduire une source de données initiale de longueur L en une autre source d'information de longueur K.

Il est clair que la longueur K doit être inférieure à L le plus possible. Donc, la compression considérée comme une fonction, doit être réversible ou presque (dans le cas de la compression avec perte).

Nous pouvons classer les méthodes de compression suivant plusieurs critères. A titre d'exemple, les algorithmes élaborés peuvent être classifiés en terme de la perte ou non de l'information, d'utilisation ou non de transformées, de la complexité d'élaboration …. [7]

II-3- Méthodes de compression sans perte

Cette catégorie de méthodes est caractérisée par la présentation exacte de l'information d'origine. Soit une source d'information $X=\{x_1, x_2, x_3, \ldots..x_n\}$. Par application d'un algorithme de compression sans perte, on aboutira à un nouveau vecteur $Y= \{y_1, y_2, y_3, \ldots..y_m)$ de telle façon que la taille du vecteur X, en bits, est nettement supérieur à celle du vecteur compressé Y.

La figure II.1 montre les compresseurs sans perte les plus utilisés en littérature spécialisée du domaine :

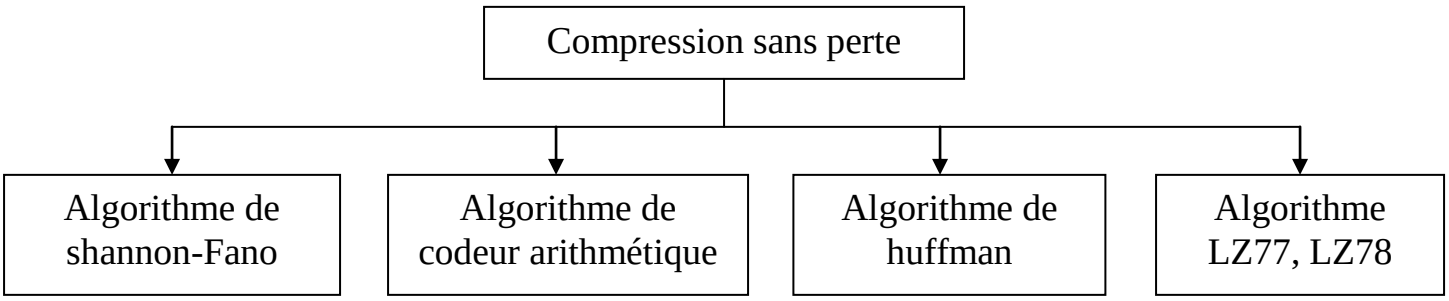


Figure II-1- Algorithmes de compression sans perte les plus utilisés

Avant d'introduire ces différents algorithmes, nous allons donner quelques définitions nécessaires.

II-3-1- Entropie d'une source d'information

Une source d'information peut être un fichier de données financières, une image, un signal Electrocardiogramme, …..

Cette source peut être modélisée par un vecteur $X=\{x_1, x_2, x_3, \ldots..x_n\}$ où x_i est une réalisation d'un code issu d'un alphabet $\{C_1, C_2, C_3, \ldots\ldots C_M\}$.

L'entropie donnant la quantité d'information moyenne est définie par [9],[10] :

$$H(X) = -\sum_{i=1}^{M} P(X = C_i)\log_2 P(X = C_i) \qquad \text{(II.1)}$$

Où $P(X=C_i)$ est la probabilité d'occurrence du code C_i dans la source d'information X

Il est à noter que la source d'information dont les codes de l'alphabet sont équiprobables c'est-à-dire, $P(X=C_1)= P(X=C_2)= P(X=C_2)=\ldots\ldots\ldots\ldots P(X=C_M)=1/M$, donne l'entropie maximale [9].

Il est notable que la compression sans perte de telle source d'information mène à l'échec. Donc, il est préférable que les fréquences d'occurrences des différents codes C_i soient distinctes et même concentrées.

II-3-2- Taux de compression

Le taux de compression (compression ratio) se trouve sous plusieurs définitions.

Cependant, la plus utilisée est [11]:

$$TC = \frac{\text{Taille de la source d'information originale (en bits)}}{\text{Taille de la source d'information compressée (en bits)}} \qquad \text{(II.2)}$$

Il est évident que l'algorithme de compression le plus performant est celui qui atteint un TC le plus grand possible.

Une autre mesure de performance, équivalente à (II.2), est le flux de bits par code (bitrate/code) [11]. Dans ce cas là, l'algorithme est le plus efficace lorsque le « bitrate » est le plus petit.

Dans la catégorie des méthodes de codage statistiques nous pouvons énumérer la méthode de Shannon-fano et celle de huffman

II-3-3- Méthode de Shannon-Fano

Le premier codage élaboré pour aboutir à des codes de tailles variables et menant à un codage moyen satisfaisant, a été celui de Shannon-Fano. Nous démarrons avec N symboles avec leurs probabilités (fréquences d'occurrences) connues, les symboles sont arrangés d'une façon descendante suivant leurs probabilités. L'ensemble des symboles est divisé en deux sous ensemble qui ont la même (ou presque la même) probabilité. Tous les symboles appartenant à un sous ensemble auront un code assigné 0 et les symboles de l'autre auront un code 1.

Chacun de ces deux sous ensembles est subdivisé de la même manière récursivement à deux sous ensembles et le bit assigné est déterminé de la même façon. Quand un sous ensemble contient juste deux symboles, leurs codes sont distingués par l'ajout d'un bit pour chacun (0 pour le premier symbole et 1 pour le deuxième symbole). [12]

Un exemple d'illustration est donné par le tableau (II.1).

La source d'information (ou le fichier contient 7 codes de probabilités {0.05, 0.10, 0.10, 0.15, 0.15, 0.20, 0.25}

Tableau II.1 Exemple de codage Shannon-Fano

Probabilité d'occurrence	étapes	Résultat de codage final (bit)
1. 0.25	1 1	: 11
2. 0.20	1 0	: 10
3. 0.15	0 1 1	: 011
4. 0.15	0 1 0	: 010
5. 0.10	0 0 . 1	: 001
6. 0.10	0 0 0 1	: 0001
7. 0.05	0 0 0 0	: 0000

Le codage moyen de cet exemple est 2.7 bits/symbole. C'est un résultat satisfaisant et proche de celui de l'entropie donnant un codage de 2.67 bits/symbole. [12]

II-3-4- Méthode de Huffman (Huffman Coding)

L'une des méthodes, communément utilisées, est l'algorithme de Huffman qui est un peu similaire au codage de Shannon-Fano depuis son invention en 1952 par D. Huffman. Ce codage a fait l'objet d'une recherche intensive liée au domaine de la compression de données [12].

L'algorithme de Huffman commence par la construction d'une liste de symboles de l'alphabet liée à une source d'information donnée d'une manière descendante relativement aux probabilités (fréquences) d'occurrence. A chaque étape les deux symboles dont les probabilités sont les plus petites sont sélectionnées et ajoutées au sommet d'un tronc partiel d'arbre, ou les deux symboles sont remplacés par le tronc.[10]

La Figure II.2 est un exemple illustratif.

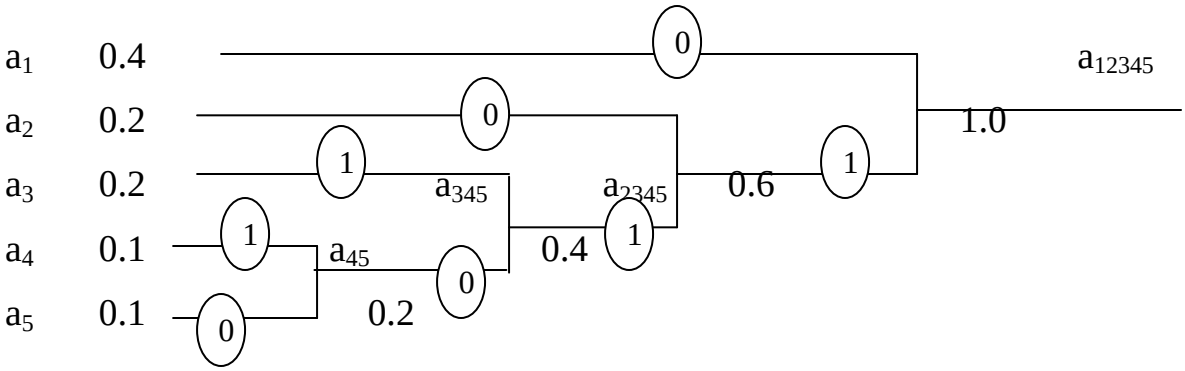


Figure II-2- Arbre de Huffman

La source d'information de cet exemple est constituée de 5 symboles de fréquences d'occurrence {0.4 (a_1), 0.2 (a_2), 0.2, (a_3) 0.1 (a_4), 0.1 (a_5)}.

- le code a_4 est groupé avec a_5 pour former un nœud a_{45} de probabilité 0.2.

- Il y a maintenant 4 symboles a_1, a_2, a_3, a_{45} de probabilités respectives {0.4, 0.2, 0.2, 0.2}. Trois symboles ont des probabilités identiques. Nous groupons arbitrairement deux d'entre eux. Nous choisissons a_{45} et a_3 pour aboutir à un nouveau code a_{345} de probabilité 0.4.
- Reste à coder, à cette étape là, trois codes {a_1, a_2, a_{345}} de probabilités respectives {0.4, 0.2, 0.4}. De même manière précédente nous obtenons a_{2345}, un nouveau code issu du groupement de a_2 et a_{345}. Ce dernier est de probabilité 0.6.
- A la fin, nous terminons par le codage des symboles restants {a_1, a_{2345}} pour arriver à 1.

L'association de bits de codage à chacun des cinq codes se fait de la manière suivante :
A chaque nouvelle étape de regroupement, nous associons un « 1 » à une branche du nœud et un zéro « 0 » à l'autre (figure II.2).
Par l'utilisation de cette stratégie de codage, nous aboutissons à un codage de tailles variables 0, 10, 111, 1101et 1100. De même, le codage moyen est de 2.2 bits/symbole. [10]

II-3-5- Méthode de codage RLE (Run-Length Encoding)

Cette méthode est destinée principalement aux sources d'information contenant fréquemment des successions de symboles similaires. Les images naturelles sont un bon exemple [10].

Dans une image les Pixels voisins sont fortement corrélés entre eux. Souvent on observe que les Pixels consécutifs dans une région douce d'une image sont identiques où la variation entre les Pixels voisins est très petite. L'aspect des codes identiques est particulièrement vrai pour des images binaires où habituellement l'image qui se compose des 0 ou des 1. Même si les Pixels consécutifs dans des images de niveaux de gris ou de couleur ne sont pas exactement identiques mais lentement changeants, ils peuvent souvent être prétraitées et les valeurs traitées consécutives de Pixels deviennent identiques. Le codage RLE est une approche simple pour coder une chaîne constituée d'une succession

d'un même code , par exemple, la donnée *d* = 5 5 5 5 5 5 5 19 19 19 19 19 19 19 19 19 19 19 0 0 0 0 0 0 0 0 8 23 23 23 23 23 23 contient des chaînes de 5, 19, 0, 23, et 8. Les données peuvent être représentées d'une manière compacte en indiquant simplement la valeur du symbole et la longueur de sa succussion (voir Figure II.3) :

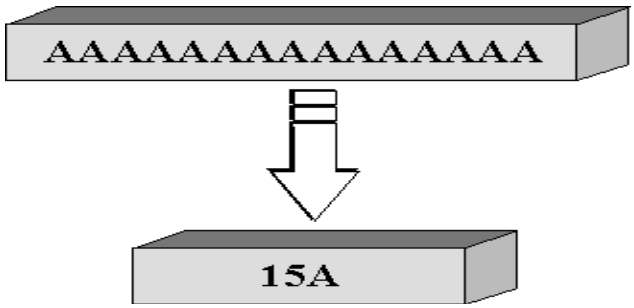


Figure II-3- Codage RLE

De cette manière, la donnée *d* peut être codée par le codage RLE comme suit : (5 7) (19 12) (0 8) (8 1) (23 6).Pour la facilité de compréhension, nous avons représenté le code sous forme des paires. Ici la première valeur représente le code, alors que la deuxième indique le nombre de répétition de ce code.

Dans certains cas, l'aspect des chaînes de symboles peut ne pas être très évident. Mais les données peuvent probablement être prétraitées afin de faciliter le codage RLE. Considérons la donnée *d* = 26 29 32 35 38 41 44 50 56 62 68 78 88 98 108 118 116 114 112 110 108 106 104 102 100 98 96. Nous pouvons simplement prétraiter cette donnée, en prenant la différence d'échantillon *e (i) = d (i) - d (i* - l), pour produire les données traitées ***t*** = 26 3 3 3 3 3 3 6 6 6 6 10 10 10 10 10 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2. Cette donnée prétraitée peut maintenant être facilement codée par RLE : (26 1) (3 6) (6 4) (10 5) (-2 11). Une variante de cette technique est appliquée dans la norme JPEG.[4]

D'autre part, les images binaires (en noir et blanc), où la valeur de chaque pixel est 0 ou 1 peuvent être aussi subir un codage RLE similaire. Par exemple, pour :

d = 000000000111111111110000000000000001110000000000000010011111111111,

le codage RLE est : *c(d)* = (0, 9, 11, 15, 3, 13, 1, 2, 10). Le premier 0 indique la valeur du premier bit, la valeur 9 indique le nombre de sa succussion, la valeur 11 indique le nombre de succussion de 1 et ainsi de suite,nous aurons une suite alternée de nombres de succussions de « 1 » ou de « 0 ».Tandis que la donnée binaire originale *d* exige 65 bits pour le stockage, sa représentation compacte *c (d)* exige 36 bits seulement dans la prétention que chaque code est représenté par 4 bits**.** [4],[13]

Il est clair qu'il y a un problème d'allocation de bits pour coder les nombres de successions

II-3-6- Codage arithmétique (Arithmetic Coding)

Dans la méthode de Huffman précédemment parlé, il est évident qu'en utilisant une longueur de code log (1/p(*x*)) correspondant *à x* est presque optimal parce qu'il a une longueur proche de 1 bit de l'entropie. Les codes optimaux jusqu'à présent sont des codes de Huffman, et ceux-ci peuvent être construits par le procédé décrit dans la section II-3-4. [2],[10]

Le codage arithmétique traite l'ensemble d'une séquence de symboles comme une seule entité, ce qui lui permet de réussir à sortir de la contrainte imposée par les approches VLC (Variable Length Coding) qui attribuent un mot de code court aux symboles probables, et des mots plus longs aux symboles moins fréquents. Ainsi, le codage arithmétique permet de représenter une séquence de symboles par un intervalle de nombres réels compris entre 0 et 1. Il est à signaler que toute valeur appartenant à ce dernier intervalle représentera d'une manière unique la séquence à coder. A mesure que la séquence s'allonge, l'intervalle requis pour le représenter diminue, et le nombre de bits qui servent à préciser cet intervalle s'accroît. Les symboles successifs d'une séquence réduisent cet intervalle en concordance avec la probabilité d'apparition du symbole. Finalement, les données comprimées consistent en la partie fractionnaire du plus court nombre binaire qui se trouve dans l'intervalle final.[14]

L'idée essentielle du codage arithmétique est de calculer efficacement p (xn) (la fonction de probabilité) et la fonction de distribution cumulative F (xn) pour le code xn de source. En utilisant les idées du codage de Shannon-Fano-Elias, nous pouvons employer un nombre dans l'intervalle [F (xn) - p (xn), F (xn)] comme code pour xn.. Par exemple, exprimer F (xn) à une exactitude du log (1/p(xn)) nous donnera un code pour la source. En utilisant les mêmes procédures que dans la discussion du code de Shannon-Fano-Elias, il suit que le mot de code (codeword) correspondant à n'importe quel ordre se trouve en dessous de l'étape dans la fonction de distribution cumulative correspondante à cet ordre. Ainsi les mots de codes pour les différents ordres de la longueur *n* sont différents. Cependant, le procédé ne garantit pas que l'ensemble de codewords soit préfixe libre. Nous pouvons construire un ensemble préfixe libre en employant F(x) arrondi au [log (1/p(xn)) +1]. Dans l'algorithme décrit ci-dessous, nous maintiendrons *F (xn)* et *p(xn)* au cours de l'algorithme, ainsi nous pouvons calculer *F(x)* facilement à n'importe quelle étape.[2]

L'algorithme du codage arithmétique statique est le suivant:

1- Soit X une séquence de m symboles $X=(x_1x_2\ldots x_m)$ qui prend ses valeurs à partir d'une source $S= \{s_1, s_2, \ldots, s_n\}$.

2- Calculer la probabilité associée à chaque symbole de la source S, notons $pi=Probabilité(S=s_i)$

3- Associer à chaque symbole sk un sous intervalle [Lsk, Hsk] proportionnel à sa Probabilité, avec (Hsk - Lsk)= pk.

4- Initialiser la limite inférieure (L) de l'intervalle de travail à la valeur 0 et la limite supérieure (H) à la valeur 1.

5- Tant qu'il reste un symbole dans la séquence à coder :

- Largeur = H - L
- L = L + largeur x Lsk

• H = L + largeur x Hsk

6- A la fin, n'importe quelle valeur de l'intervalle [L,H] représente d'une manière unique la séquence d'entrée.

Généralement, on choisit comme représentant de la séquence la limite inférieure (L) de l'intervalle, ou bien on peut choisir la valeur moyenne: [(L+H)/ 2]. On appelle code(X) la valeur choisie comme représentant de la séquence à coder.

Il est à noter qu'à la fin du processus de codage, le codeur arithmétique génère un fichier composé par un en-tête suivi de la représentation binaire du code choisi. De plus, l'en-tête peut contenir soit la table des probabilités, soit la table des fréquences de tous les symboles.

Après le codage de la séquence X, on peut facilement décoder cette séquence en utilisant la table des probabilités (ou bien la table des fréquences) et en appliquant l'algorithme suivant :

1- D'abord on doit initialiser la limite inférieure (L) de l'intervalle de travail à la valeur 0 et la limite supérieure (H) à la valeur 1.

2- Tant qu'il reste un symbole à décoder :

- largeur = H – L
- Chercher le sous intervalle [Lsk, Hsk] du symbole à décoder sachant que :

  $Lsk \leq (code(X) - L) / largeur \leq Hsk$
- décoder le symbole sk
- L = L + largeur x Lsk
- H = L + largeur x Hsk

3- A la fin, on arrive à décoder la séquence X.[13][2]

Il est à signaler que le codage arithmétique statique utilise une table des probabilités fixe durant les processus de codage et de décodage. De plus, on peut calculer, à partir de la table des probabilités, la probabilité de la séquence p(X) qui est égale à :

$$p(X) = p_m.p_{m-1}.....p_2.p_1 = \prod_{i=1}^{m} p_i \qquad \text{(II-3)}$$

Après l'étape de codage, la séquence X sera représentée par un intervalle I= [L,H]. L'intervalle I peut s'écrire sous la forme I= [L, L+v] avec v = H-L : la largeur de l'intervalle. Ainsi, d'après l'algorithme de codage, on peut facilement déduire que la largeur de l'intervalle v est égale à la probabilité p(X) de la séquence X. Par conséquent, le code(X) peut être représenté uniquement avec un nombre de bits qui est égale à la valeur :

$$\log_2 \frac{1}{p(X)} + 1 = -\log_2 p(X) + 1 \qquad \text{(II-4)}$$

Ainsi, si la largeur de l'intervalle requis pour représenter une séquence X diminue, alors le nombre de bits qui servent à préciser cet intervalle s'accroît.

Exemple1

Soit, par exemple, à coder la séquence X= " SSSSSWWWIII ". En appliquant l'algorithme du codage arithmétique sur la séquence X, on obtiendra L'intervalle suivant [0.02252875, 0.0225625].

$$P(X) = \frac{5}{10}.\frac{5}{10}.\frac{5}{10}.\frac{5}{10}.\frac{5}{10}.\frac{3}{10}.\frac{3}{10}.\frac{3}{10}.\frac{2}{10}.\frac{2}{10} = 0.00003375$$

De plus, nous avons besoin au minimum de : [- $\log_2$p(x) +1]=16 bits pour coder cette séquence.[14]

Exemple 2

On veut coder la séquence suivante (acaab) avec la probabilité d'apparition de chaque caractère est donnée dans le tableau II-2 [15]

Tableau II-2- Probabilités des symboles

symbole	probabilité	intervalle
a	0.7	[0, 0.7]
b	0.1	[0.7, 0.8]
c	0.2	[0.8, 1.0]

La figure II-4- donne une représentation du processus de codage arithmétique de la chaîne « acaab

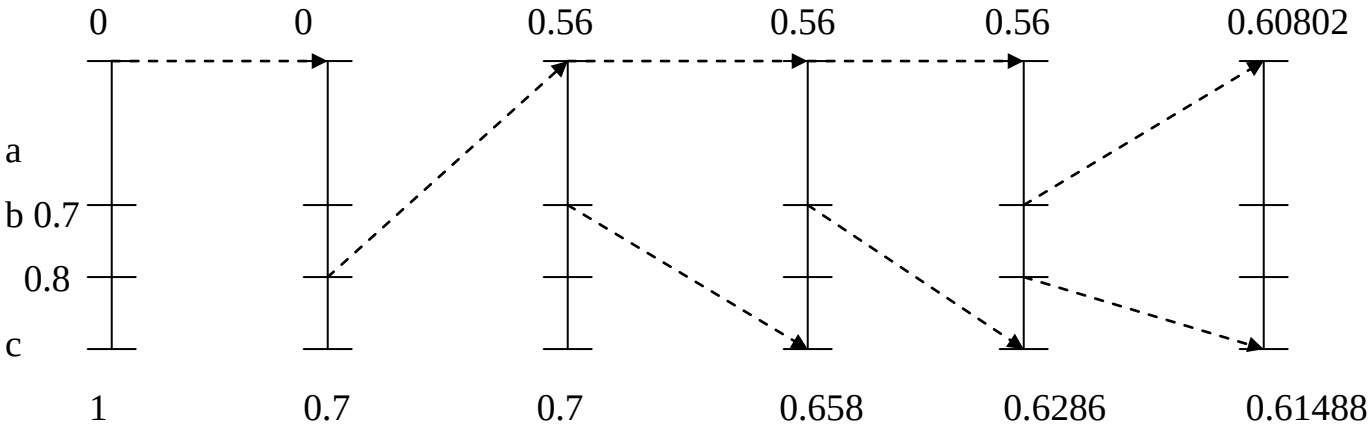


Figure II-4- Génération de l'étiquette pour la séquence « acaab »

II-3-7- Méthodes de dictionnaire (Dictionary Methods)

Dans les méthodes de codage citées précedament, la compression sans perte a été obtenue en employant un modèle de probabilité pour induire un codeur. Les méthodes de dictionnaire emploient une liste d'expressions (un dictionnaire), où plusieurs expressions dans la source, sont remplacées par des indicateurs. La compression se produit si les

indicateurs exigent moins d'espace que les fragments correspondants.(Naturellement, la méthode de passer le dictionnaire doit également être considérée.)[19]

De beaucoup de manières, il est plus facile de comprendre les méthodes à base de dictionnaire. Au niveau le plus simple, plusieurs dictionnaires spécialisés (fixés) pourraient être rendus disponibles au codeur et au décodeur. Au texte en anglais, quelques milliers de mots les plus généralement utilisés pourraient servir de dictionnaire ;si la source se composait du code en certain langage de programmation tel que le C, une liste des mots-clés et des fonctions de bibliothèque standard pourrait servir comme mots de dictionnaire. Les dictionnaires fixes peuvent être utiles dans quelques situations, mais il y a au moins deux inconvénients sérieux. D'abord, les dictionnaires doivent êtres connus au codeur et au décodeur. Des changements au dictionnaire devraient être propagées à tous les emplacements qui emploient l'arrangement. En second lieu, les arrangements fixes de dictionnaire ne peuvent pas comprimer le texte "inconnu".Dans le cas du code de C, il y aurait probablement peu de compression des noms de variables créés par le programmeur.
Notre intérêt principal ici est lié aux méthodes qui s'adaptent à la source ;c'est-à-dire, méthodes qui construisent le dictionnaire de la source, et qui font habituellement ceci pendant que la source est balayée. [19][10]

A la fin des années 70, deux chercheurs, Jacob Ziv et Abraham Lempel, mettent au point une technique de compression encore plus sophistiquée, appelée compression LZ (d'après leurs initiales). Ils inauguraient la compression moderne, exploitant un dictionnaire mobile sur le même principe que la compression arithmétique. C'était, depuis Huffman, la première fois que l'on mettait sérieusement en doute l'efficacité des méthodes statistiques. Le principe est assez simple à comprendre: pour coder un mot, on peut effectivement le transmettre, où ne transmettre que sa position dans un dictionnaire contenant tous les mots possibles. Dans le deuxième cas, il n'y a qu'un entier à transmettre. L'inconvénient est qu'il faut disposer d'un dictionnaire exhaustif. L'idée maîtresse de LZ et des algorithmes à dictionnaire est de créer ce dictionnaire pendant le processus de compression.[19]

Deux procédés en dérivent, connus sous les noms de LZ77 et LZ78, établis d'après leurs années de création. Ils reposent sur l'enregistrement de séquences de caractères dans une sorte de dictionnaires de "phrases", édifié à partir des données en entrée, au fur et à mesure de la lecture du fichier à compacter. Le texte qui reste à compresser est comparé au contenu du dictionnaire. Dès qu'il rencontre une nouvelle séquence, l'algorithme de compression vérifie sa présence dans le dictionnaire. Si elle ne s'y trouve pas, il l'ajoute dans le dictionnaire et place en sortie un symbole marqueur qui en identifie l'adresse dans une table réservée à cet usage. Si la table a déjà été enregistrée, le compresseur se contente de mettre la table à jour, en y reportant l'identifiant.[16]

II-3-7-1- LZ77

Le premier algorithme de compression décrit par Ziv et Lempel désigné généralement sous le nom de LZ77. Elle utilise un dictionnaire qui n'est pas stocké dans le fichier compressé qu'elle construit dynamiquement, au cours de la compression et de la décompression. Plus compliqué que le RLE, il s'agit cette fois de repérer des motifs répétés composés de séries variables en longueur de bits ou d'octets. Chaque motif est copié dans le dictionnaire et se voit attribué un indice. Puis chaque motif est remplacé dans le fichier par son indice, sachant qu'une valeur isolée n'est pas codée. Elle a besoin d'un apprentissage pour être efficace, pour reconnaître des longues chaînes répétées. En effet, l'algorithme ne fonctionne pas sur un nombre fixe de motifs mais apprend les motifs du fichier durant la lecture du fichier à compacter. Elle est donc peu efficace sur des petits fichiers. Cette méthode est très rapide. [7],[12]

II-3-7-2- LZ78

Elle construit son dictionnaire comme un "arbre de phrases" ou chaînes auquel un caractère spécial est ajouté pour chaque séquence rencontrée en cours de compression. Cette dernière se fonde donc sur le remplacement d'une chaîne de texte longue par une référence chiffrée plus courte. Il faut partir d'un dictionnaire vide puis, pour compacter une séquence, rechercher la phrase la plus longue trouvée dans le dictionnaire, encoder en sortie la référence chiffrée trouvée dans le dictionnaire ainsi que le caractère qui la suit, ajouter au dictionnaire une nouvelle phrase constituée de la chaîne de caractère vérifiée combinée avec le caractère et continuer jusqu'à la fin du fichier..[19][10]

II-3-7-3- LZW

En 1984, pendant qu'il travaillait chez Unisys, Terry Welch modifia l'algorithme de compression LZ78 pour implémenter des contrôleurs de disques à hautes performances. Il observa que le caractère littéral suivant le numéro de "phrase" en sortie pouvait être éliminé et la compression améliorée. L'idée était de placer d'office dans le dictionnaire 256 phrases initiales d'un caractère chacune. Tout ce qui resterait à faire serait, en sortie, d'écrire la référence de la phrase sans l'accompagner d'un littéral. Ce dernier serait simplement considéré comme le premier caractère de la phrase suivante, et le dictionnaire serait incrémenté d'une manière naturelle (avec élimination, quand le dictionnaire est saturé, des mots les moins utilisés)[18]

Ce principe forme la base de la compression LZW (Lempel-Ziv-Welch). La méthode est semblable à la méthode RLE, mais elle est appliquée à des suites d'octets. Elle connut un succès immédiat et fut largement suivie aussi bien dans le monde UNIX que dans le monde DOS.

La force de LZW étant que c'est un algorithme de compression capable de travailler sur presque n'importe quel type de données. Il est généralement rapide aussi bien en phase de

compression que de décompression et ne requière pas l'utilisation d'instructions en virgule flottante.[10]

En ce qui suit, on donne un exemple illustré par la figure II.5.

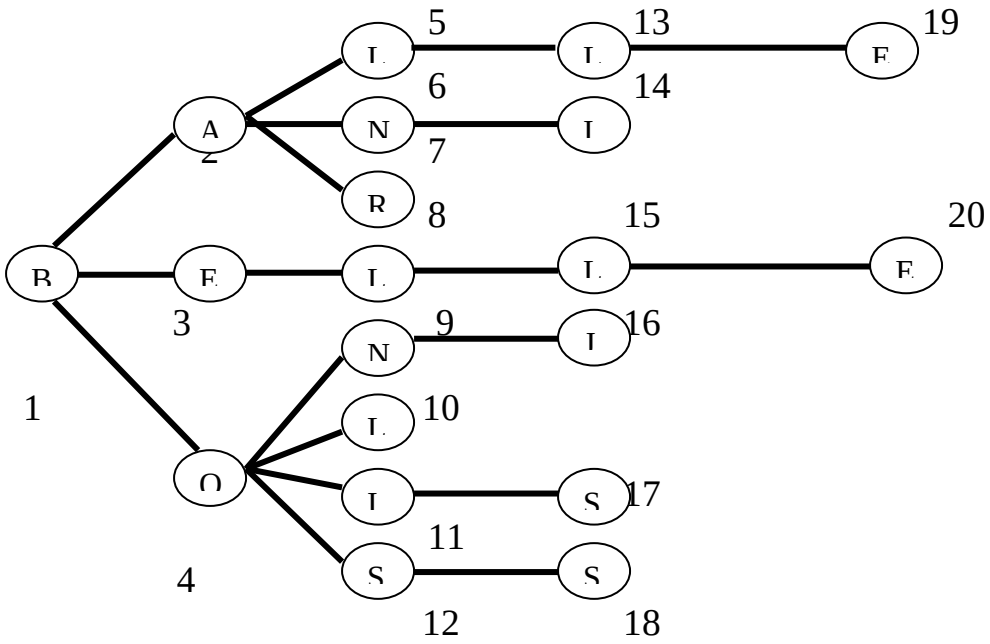


Figure II-5- Exemple de codage LZW

Lorsqu'on veut encoder un fichier, on cherche à obtenir la plus grande chaîne se trouvant déjà dans l'arbre pour la remplacer par son indice. Lorsque la chaîne est absente, on ajoute un nouveau noeud à l'arbre qui pourra servir ultérieurement jusqu'a ce que le fichier soit lu en entier ou que le dictionnaire soit plein.

Pour décoder un indice dans un dictionnaire sous forme d'arbre il suffit donc de commencer par l'indice et de remonter de nœud en nœud jusqu'à sa source puis ensuite inverser la chaîne obtenue pour voir la chaîne originale.

Ainsi donc :

- On part de l'indice 20, nous avons la lettre < E>.

- On remonte a son parent a l'indice 15 on se trouve la lettre < L >.
- On remonte encore a son parent a l'indice 8 on nous trouvons la lettre <L >.
- On remonte encore a son parent a l'indice 3 on nous trouvons la lettre <E >.
- On remonte encore a son parent a l'indice 1 on nous trouvons la lettre <B >.
- On obtient donc la chaîne < ELLEB >.
- Finalement, on inverse la chaîne obtenue, ce qui nous donne le mot < BELLE >.

II-4- Méthodes de compression avec perte

Dans la compression sans perte (Lossy Compression) le taux de compression n'est pas toujours suffisant pour la transmission et le stockage de données de grandes tailles, et pour descendre au dessous de l'entropie on agit sur la source d'information avec une méthode de compression avec perte mais ça entraîne le dilemme (taux de compression – distorsion). [8]. Pour l'améliorer, il va falloir accepter quelque dégradation. Ceci permet d'atteindre des taux de compression arbitrairement grand au prix d'une dégradation toujours plus importante. L'objectif des algorithmes de compression avec perte est de minimiser cette dégradation de qualité pour un taux de compression donné. Donc la clé de la compression avec perte est dans une modification non réversible de la source permettant d'obtenir une nouvelle source dont l'entropie est plus faible. Il existe deux catégories principales de compression avec perte :

- Les méthodes directes
- Les méthode par transformées

La figure [II-6] présente un schéma synoptique des méthodes de compression avec perte :

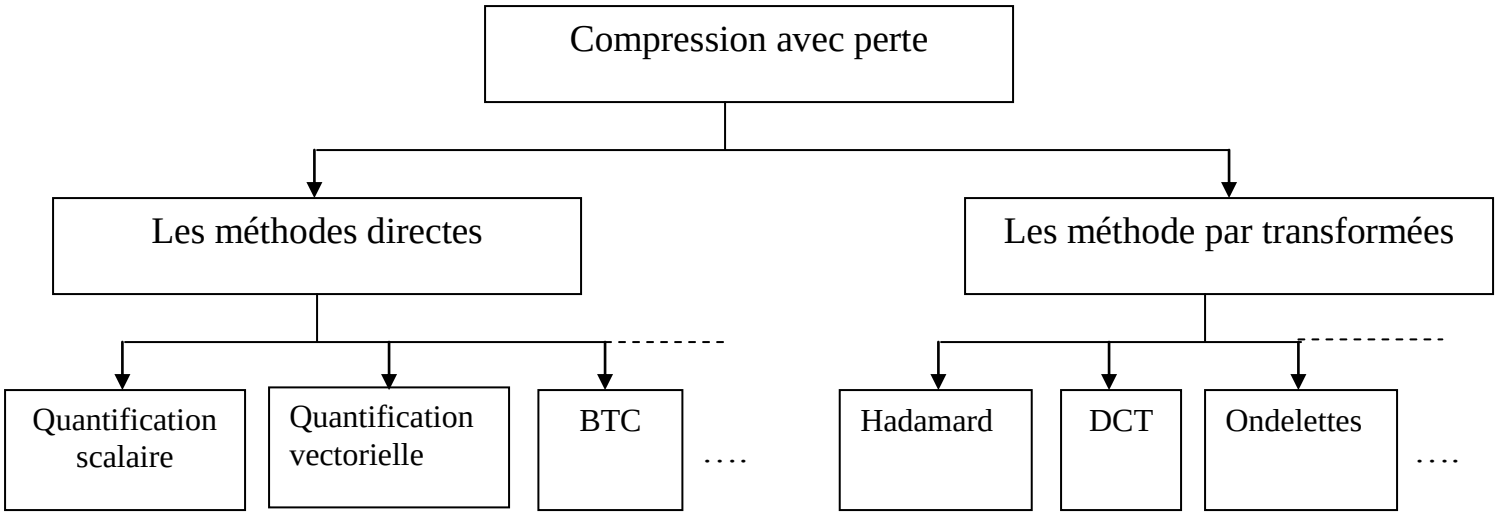


Figure II-6- Méthodes de compression avec pertes

II-4-1- Mesures de performances

Comme n'importe quel système, la mesure des performances d'un algorithme de compression de données est un critère important pour le choix de l'algorithme. Les mesures de performances liées à une méthode de compression de données peuvent être regardées dans des différentes visions selon les conditions d'application : la quantité de compression, qualité objective et subjective des données reconstruites, complexité relative de l'algorithme, vitesse d'exécution, etc.…Nous expliquons certains d'entre eux ci-dessous.

II-4-1-1- Taux de compression (TC) et Bit par symbole (bite/symbole)

Le critère le plus populaire de la mesure de performance d'un algorithme de compression de données est le Taux de compression (**compression ratio).** Il est défini comme le rapport du nombre de Bits pour représenter les données originales au nombre de Bits pour représenter les données comprimées. Considérer une image de la taille 256 x 256 exigeant 65536 octets pour les stockée si chaque Pixel est représenté par un octet, et i la version

comprimée de l'image peut être stockée en 4096 bytes, le rapport de compression réalisé par l'algorithme de compression sera 16:l.

Une variante du taux de compression est le Bit par symbole (**Bits per Sample**). Ce critère indique le nombre moyen de Bits pour représenter un symbole de données (par exemple, *Bit par Pixel* pour le codage d'image).Si 65536 Pixels (un octet /pixel) d'une image sont comprimés à 4096 octets, nous pouvons dire que l'algorithme de compression a réalisé 0.5 bit par Pixel en moyenne.

Si le même algorithme est appliqué dans un certain nombre de fichiers de données distincts, l'algorithme rapportera de différents taux de compression . Par conséquent le Bit par symbole qui peut être réalisé dans la compression presque sans perte est limité par l'entropie du fichier de données selon le théorème de Shannon. Les sources avec moins de redondance ont plus d'entropie et par conséquent sont plus difficiles à compresser. [11]

II-4-1-2- Mesure de qualité

Ce critère n'est pas approprié pour des algorithmes de compression sans perte. La qualité est particulièrement importante pour des algorithmes de compression avec perte pour la vidéo, l'image, la voix, etc., parce que la donnée reconstruite est différente de l'originale et l'être humain est le juge final de la qualité reconstruite. Par exemple, s'il n'y a aucune différence perceptible entre les données reconstruites et les données originales, l'algorithme de compression peut être prétendu réaliser une qualité très haute ou la fidélité élevée. La différence entre les données reconstruites et originales s'appelle la déformation (dégradation). On compte avoir plus de haute qualité des données reconstruites, si la déformation est minimale. Les mesures de qualité peuvent être subjectifs en se basant sur la perception humaine ou peuvent être objectivement définies en utilisant l'évaluation mathématique ou statistique. Bien qu'il n'y ait aucune mesure universellement admise de la mesure de qualité, il y a des mesures objectives et subjectives :[11]

a- Mesure de qualité subjective (MOS)

La qualité subjective est définie comme un point moyen d'observateurs, parfois, s'appelle également les points **moyens d'opinion**. Il y a différentes manières statistiques de calculer le MOS. Dans l'une des manières les plus simples, un nombre statistiquement significatif d'observateurs sont aléatoirement choisis pour évaluer la qualité visuelle des images reconstruites. Toutes les images sont comprimées et décomprimées par le même algorithme. Chaque observateur assigne des points numériques à chaque image reconstruite basée sur sa perception de la qualité d'image, les opinions sont classés dans une marge de 1 à 5 réponses, pour décrire par exemple la qualité de l'image, niveau 5 étant la plus haute qualité et le niveau 1 étant la plus mauvaise qualité. La moyenne des points assignés par tous les observateurs aux images reconstruites s'appelle **les points moyens d'observateur (MOS**: Mean of Scores**)** et il peut être considéré comme étant une mesure subjective. [11]

b- Mesure de qualité objective

Il n'y a aucune mesure universellement admise pour la qualité objective des algorithmes de compression de données. Les mesures objectives (Objective Quality Metrics) les plus largement répandues de qualité sont : la racine de l'erreur quadratique moyenne (root-mean-squared-error (RMSE)), le rapport signal/bruit *(S N R)*, et le rapport signal/bruit maximal *(PSNR)*. Si **I** une image *de M X N* et $\bar{I}$ l'image reconstruite correspondante après compression et décompression, *RMSE* est calculée comme suit :

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I(i,j)-\bar{I}(i,j)\right]^2} \qquad \text{(II-5)}$$

Où i, j se reportent à la position du Pixel dans l'image. *Le S N R* dans l'unité est le décibel (dB) est exprimé comme suit :

$$SNR = 20\log_{10}\left(\frac{\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}I^2(i,j)}}{RMSE}\right) = 10\log_{10}\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}I^2(i,j)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I(i,j)-\bar{I}(i,j)\right]^2}\right) \quad \text{(II-6)}$$

Dans le cas d'une image de 8 bits, le PSNR correspondant en dB est calculé comme suit :

$$PSNR = 20\log_{10}\left(\frac{255}{RMSE}\right) \quad \textbf{(}\text{II-7)}$$

Où le 255 est la valeur maximum possible du Pixel (cas des images 8 bits/pixel).

Il convient à noter qu'un RMSE petit (ou d'une manière équivalente, un S N R ou un PSNR plus élevé*)* n'indique pas nécessairement toujours une qualité subjective plus élevée. Cette mesure objective d'erreur ne se corrèle pas toujours bien avec la mesure subjective de qualité. Il y a beaucoup de cas où le PSNR d'une image reconstruite peut être raisonnablement haut, mais la qualité subjective est vraiment mauvaise une fois visualisée par les yeux humains. Par conséquent le choix de la mesure objective ou subjective pour évaluer un algorithme de compression et de décompression dépend souvent des critères d'application. [11]

II-4-2- Méthodes directes

II-4-2-1- Quantification

La quantification (Quantization) consiste à remplacer un ensemble continu de valeurs par des valeurs discrètes. Dans le domaine de la compression de données, la quantification est utilisée dans deux contextes comme suit :

1- Si les symboles de données sont des nombres flottants, la quantification peut arrondir chacun au un nombre entier le plus proche. Si les symboles de données sont de grands nombres, la quantification peut les convertir en petits nombres. Les petits nombres

prennent moins d'espace que les grands, donc une compression produite par la quantification. D'autre part, les petits nombres donnent moins d'information que les grands, qui implique une compression avec perte.

2- Si les données à compresser sont analogiques (comme une tension qui change avec du temps), la quantification est utilisée pour les digitaliser en nombres (normalement entiers).Ceci est désigné sous le nom de la conversion (ANALOGIQUE-NUMERIQUE). [12,10]

Considérons une source qui produit des nombres entre -10.0 et 10.0.Un arrangement simple de quantification devrait représenter chaque code de la source avec la valeur de nombre entier la plus proche de lui. (Si le code source est également de près de deux nombres entiers, nous sélectionnerons aléatoirement l'un d'entre eux.)Par exemple, si le code source est 2.47, nous le représenterions en tant que 2, et si le code source est 3.5415926, nous le représenterions en tant que 4.

Cette approche réduit la taille de l'alphabet exigé pour représenter le code source, le nombre infini de valeurs entre [-10.0 et 10.0] sont représentés avec un ensemble qui contient seulement 21 valeurs ({- 10.....0.....10}). Cependant, nous avons perdu pour toujours les valeurs originales (de quelques codes de la source). Si la valeur de reconstruction est 3, nous ne pouvons pas dire si le code source était 2.95, 3.16, 3.057932, ou tout autre d'un ensemble infini de valeurs. En d'autres termes, nous avons perdu de l'information. Cette perte d'information est la raison pour l'usage du mot avec perte dans la compression de données.[20][17]

Les d'entrées et les sorties d'un quantificateur peuvent être des grandeurs scalaires ou des vecteurs. S'ils sont scalaires, le quantificateur générateur est dit quantificateur scalaire. S'ils sont des vecteurs, elles indiquent un quantificateur vectoriel.

II-4-2-1-a- Quantification scalaire SQ

La quantification scalaire (Scalar Quantization) est un processus très simple. Cependant, la conception du quantificateur a un impact significatif sur la quantité d'information compressée obtenue et sur la perte résultante. Par conséquent, nous consacrerons beaucoup d'attention aux issues liées à la conception des quantificateurs.

Dans la pratique, le quantificateur se compose de deux parties : la première relative à l'encodeur et la seconde spécifiant le décodeur. L'encodeur divise la gamme des valeurs issues de la source produite dans un certain nombre d'intervalles. Chaque intervalle est représenté par un codeword distinct. L'encodeur génère tous les codewords relatifs *chacun* à un intervalle. Quand les valeurs de la source d'information viennent d'une source analogique, l'encodeur s'appelle un convertisseur (ANALOGIQUE-NUMÉRIQUE)[10]

L'encodeur pour un quantificateur avec huit valeurs de reconstruction est montré sur la figure II-7. Dans cet exemple, tous les symboles qui ont des valeurs entre -1 et 0 seraient assignés le code 011. Tous les symboles entre 0 et 1.0 seraient assignés le code 100, et ainsi de suite. Sur les deux frontières, toutes les entrées plus grandes que la valeur 3 auront une assignation de code 111, et toutes les entrées avec les valeurs moins de -3.0 seraient assignées le code 000. [20, 10, 12]

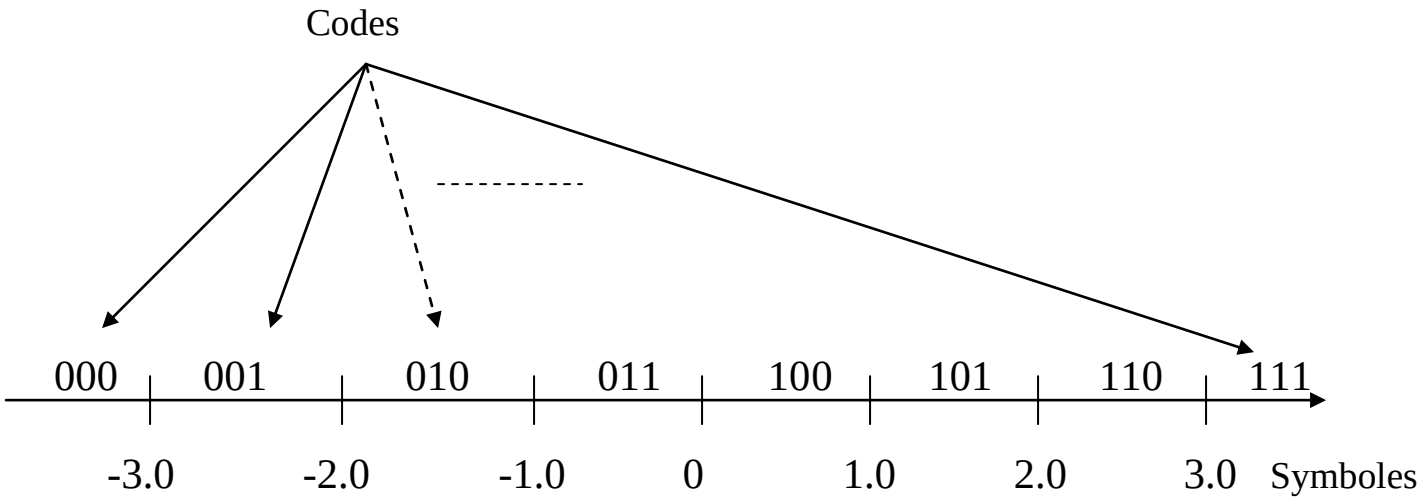


Figure II-7- Schéma d'un quantificateur 3-bits[20].

- **Quantification uniforme**

Le type le plus simple de quantificateurs est le quantificateur uniforme. Tous les intervalles sont de même taille, excepté probablement pour les deux intervalles externes. En d'autres termes, les frontières de décision sont espacées également. Les valeurs de reconstruction sont également espacées, avec le même espacement que les frontières de décision. Par conséquent, dans les intervalles intérieurs, elles sont les points médians des intervalles. Le quantificateur uniforme est représenté sur la figure II-8 .[20, 12]

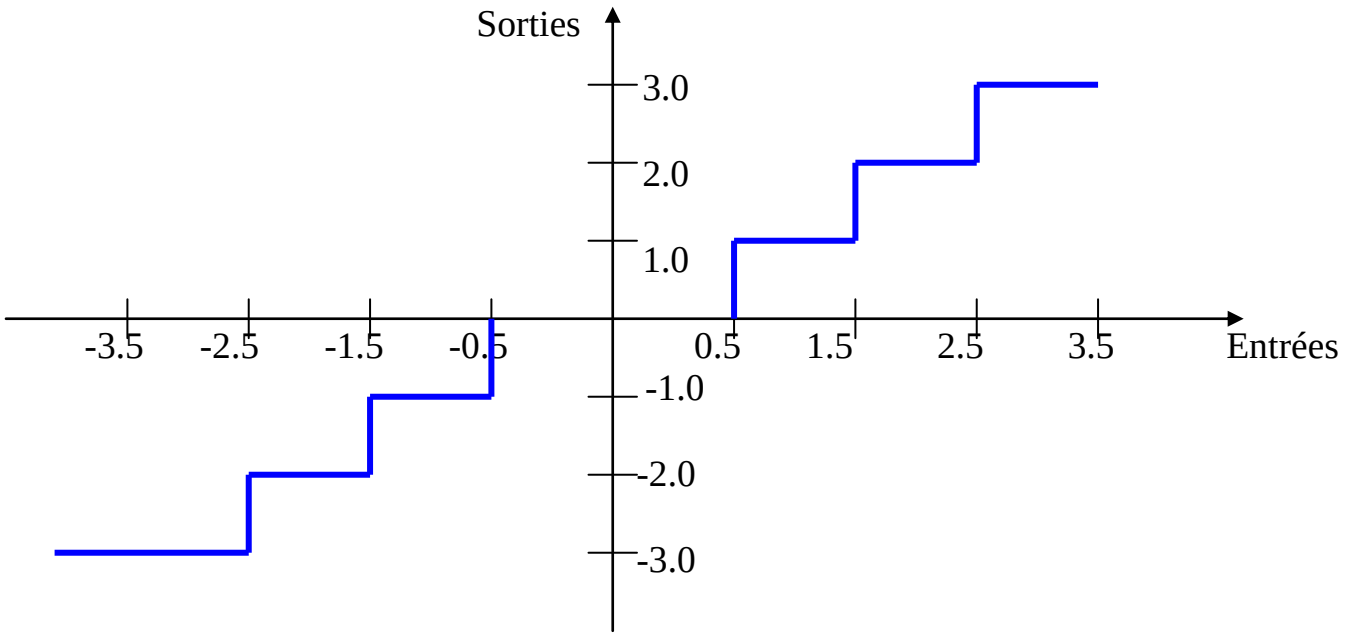


Figure II-8- Quantificateur uniforme

En ce qui suit, nous donnons un exemple d'utilisation du quantificateur scalaire uniforme dans la compression d'image :

Un modèle de probabilité pour les variations de Pixels dans une image est impossible presque à obtenir en raison de la grande variété d'images disponibles. Une approche commune est de déclarer les valeurs de Pixel pour être uniformément distribuées entre 0 et 2^n - 1, où *n* est le nombre de bits par Pixel.Pour la plupart des images nous supposons le cas du nombre de bits par Pixel est 8. Donc, on assumerait que les valeurs de Pixel

changent uniformément entre 0 et 255.Quantifions notre image d'essai en utilisant un quantificateur uniforme.

Si nous voulions employer seulement 1 bit par Pixel, nous diviserions la gamme [0, 255] en deux intervalles : [0, 127] et [128, 255].Le premier intervalle serait représenté par la valeur 64, le point milieu du premier intervalle, tandis que les Pixels dans le deuxième intervalle seront représentés par la valeur 196. En d'autres termes, les limites d'intervalles sont {0, 128.255}, alors que les valeurs de reconstruction sont {64, 196}.L'image quantifiée est montrée sur la figure II-9. Comme prévu, presque tous les détails dans l'image ont disparu. Si nous devions employer un quantificateur 2-bits, avec les valeurs {0, 64, 128, 196, 255} et les niveaux de reconstruction {32, 96.160, 224}, nous obtenons considérablement plus de détail. Le niveau du détail augmente à mesure que l'utilisation du bits augmente jusque à 6 bits par Pixel, l'image reconstruite est indistinguable de l'original, au moins à un observateur occasionnel. [20-21]

Figure II-9- L'image en haut à gauche « originale » en haut à droite 1bit/pixel
En bas à gauche 2 bit/pixel en bas à droite 3bit/pixel

- **Quantification non uniforme**

Comme nous pouvons voir du la figure II-10. Afin de diminuer la déformation moyenne, nous pouvons essayer de mieux approximer l'entrée dans les régions de la probabilité élevée, peut-être au coût de plus mauvaises approximations dans les régions de la probabilité inférieure. Nous pouvons faire ceci en rendant les intervalles de quantification plus petits dans les régions qui ont plus de probabilité. Si la distribution de source est comme la distribution représentée sur la figure II-10, nous aurions de plus petits intervalles près de l'origine. Un quantificateur qui a des intervalles non uniformes s'appelle un quantificateur non uniforme. Un exemple d'un quantificateur non uniforme est montré sur la figure II-10 [21]

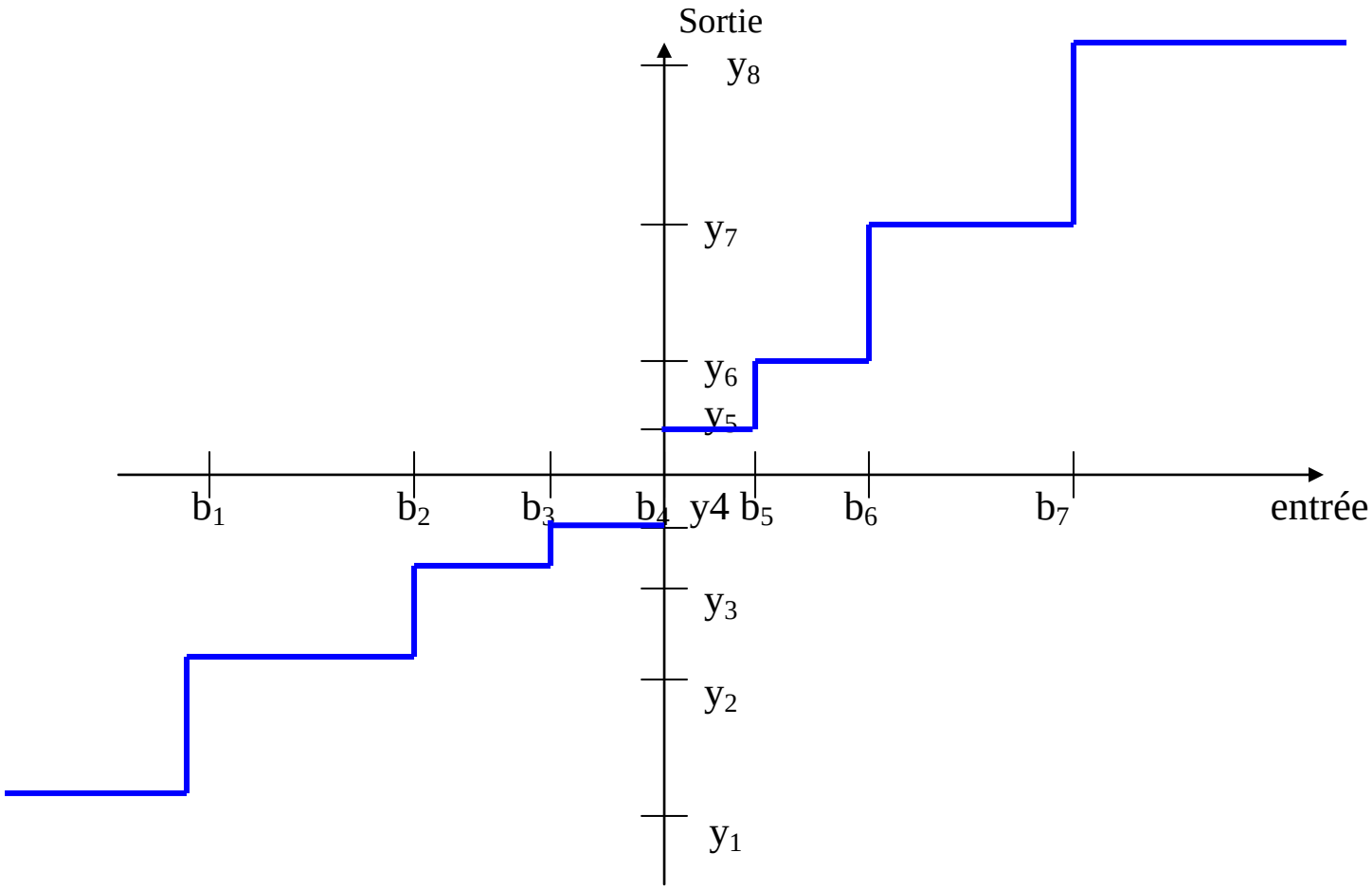


Figure II-10- Quantificateur non uniforme

II-4-2-1-b- Quantification vectorielle QV (Vector Quantization)

La quantification vectorielle (QV) consiste à représenter tout vecteur x_i de dimension k par un autre vecteur y_i de même dimension, mais ce dernier appartenant à un ensemble fini μ de L vecteurs. Les vecteurs *yi* sont appelés vecteurs représentants, vecteurs de reproduction où code vecteurs. μ est le dictionnaire ou le catalogue des formes. [17]

Dans la quantification vectorielle nous groupons les codes source dans des blocs ou des vecteurs. Par exemple, nous pouvons traiter *n* symboles consécutifs de la parole comme composants d'un vecteur *n*-dimensionnel, Ou nous pouvons encore prendre un bloc de *n* Pixel d'une image et considérer chaque valeur de Pixel comme une composante d'un vecteur. Cette taille ou dimension des sorties de source forme l'entrée au quantificateur vectoriel À l'encodeur et au décodeur du quantificateur vectoriel, nous avons un ensemble de vecteurs n-dimensionnels appelés le dictionnaire du quantificateur vectoriel. Les vecteurs dans ce dictionnaire, connus sous le nom de code-vecteurs, sont choisis pour être représentant des vecteurs que nous produisons du code source. Chaque code-vecteur est assigné un index binaire. À l'encodeur, le vecteur d'entrée est comparé à chaque code-vecteur afin de trouver le code-vecteur le plus proche du vecteur d'entrée. Les éléments de ce code-vecteur sont les valeurs quantifiées, par la suite, nous transmettons ou stockons l'index binaire du code-vecteur. Puisque le décodeur a exactement le même dictionnaire, il extrait à partir de l'index binaire le code vecteur associé. Une représentation de ce processus est montrée sur la figure II-11 Bien que l'encodeur puisse devoir exécuter une quantité considérable de calculs afin de trouver le vecteur de reproduction le plus étroit au vecteur de sortie de source, le décodage se résume en une simple consultation de table. [21-20]

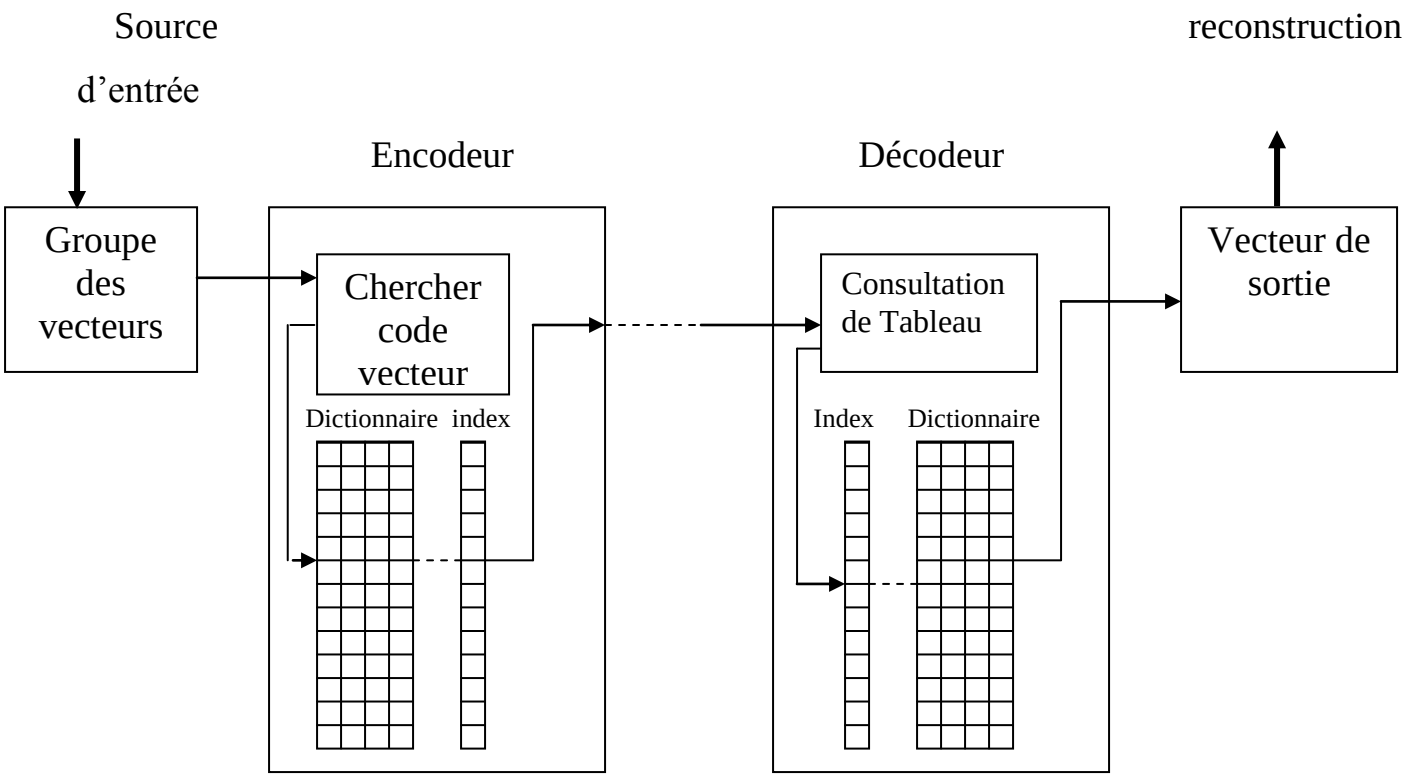


Figure II-11- Procédure de quantification vectorielle

De nombreux algorithmes de génération du dictionnaire ont été proposés : LBG, K-MEANS, KOHONEN, et l'algorithme d'apprentissage à compétition (AC).nous développant dans qui suit l'algorithme de base qui est l'algorithme LBG [20]:

- **Algorithme (LBG) (Linde-Buzo-Gray)[20]**

Nous nous concentrons sur l'un des éléments les plus importants dans la conception d'un quantificateur vectoriel, qui est la génération du dictionnaire **(**codebook). Tandis qu'il y a un certain nombre de manières d'obtenir le dictionnaire de quantificateur vectoriel, la plupart des algorithmes sont basées sur une approche particulière, populairement connue sous le nom d'algorithme (LBG) (Linde-Buzo-Gray).

Cet algorithme connu sous le nom de LBG est une généralisation de la méthode proposée par « Lloyd » en 1960, pour la quantification scalaire. Il a pour but de générer une partition sur une image (séquence d'apprentissage), partant d'un dictionnaire initial composé de vecteurs les plus éloignés possibles. Ces vecteurs doivent êtres représentatifs des vecteurs rencontrés parmi les images à coder. L'algorithme itératif converge vers un

dictionnaire localement optimal.[20]

- Détail de l'algorithme

Etape 1 :

On génère un dictionnaire initial C^0 composé de N vecteurs Y_i(i =1…N), une mesure de distorsion d, un seuil ε≥0, un compteur d'itération l = 0, une distorsion moyenne D initialisée avec une très grande valeur et une séquence d'apprentissage composée de n vecteurs (X_j , j = 1…n), D_m = valeur maximale, p : nombre maximum d'itérations.

Etape2 :

Cherchant les décisions a la limite

$$b_j^k = \frac{y_{j+1}^k - y_j^k}{2} \qquad \text{j = 1, 2, ……N-1} \qquad \text{(II-8)}$$

Étape 3 :

Calculant la distorsion D

$$D^{(k)} = \sum_{i=1}^{N} \int_{b_{i-1}^{(k)}}^{b_i^{(k)}} (x - y_i)^2 \, f_x(x) dx \qquad \text{(II-9)}$$

$f_x(x)$ *est la fonction de distribution*

Etape 4 **:**

Si $D^{(k)} - D^{(k-1)} \langle \varepsilon$, stop, Si non, continue

Etape 5 :

Incrémenter k (k=k+1), et calculer le nouveau vecteur reconstruit

$$y_j^{(k)} = \frac{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} x f_x(x) dx}{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} f x(x) dx} \qquad \text{(II-10)}$$

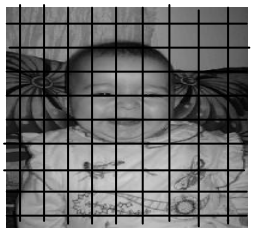
Aller a l'étape 2

II-4-2-1-c- BTC (Block Truncation Coding) [22]

La BTC est un algorithme de compression pratiquement simple, conçu pour la compression des images niveaux de gris. Son principe est de diviser l'image originale en blocs(typiquement, 4×4), puis utiliser un quantificateur pour réduire le nombre de niveaux de gris dans chaque bloc, tout en conservant la même moyenne et écart-type

- **Algorithme de codage BTC**

1- Décomposer l'image à des bloc de nxn (pratiquement 4x4).



2- Calculer la moyenne ($\overline{X}$) et l'écart type (σ), pour chaque bloc de pixels, tels que :

La moyenne $\overline{X} = \frac{1}{m}\sum_{i=1}^{m} x_i$ (m = n x n) (II-11)

L'écart type $\sigma^2 = \left(\overline{X^2}\right) - \left(\overline{X}\right)^2$ (II-12)

Avec $\overline{X^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2$ (II-13)

3- construire un nouveau bloc de même taille que celle du bloc de l'image à quantifier, puis remplir ce bloc par des 0 et des 1 de telle sorte que, si la valeur du pixel correspondant est supérieur à $\overline{X}$, alors la valeur égale à 1, sinon la valeur du pixel égale à 0.

$$Y(i,j) = \begin{cases} 1 & X(i,j) > \overline{X} \\ 0 & X(i,j) \le \overline{X} \end{cases} \quad \text{(II-14)}$$

4- calcul des niveaux de reconstruction (A et B) tel que :

$$A = \overline{X} - \sigma\sqrt{\frac{q}{m-q}} \quad , \quad B = \overline{X} + \sigma\sqrt{\frac{m-q}{q}} \tag{II-15}$$

Où σ est l'écart type, m est le nombre total de pixels dans le bloc et q est le nombre de pixels supérieur à la moyenne $\overline{X}$.

5- sauvegarder A et B et le bloc binaire Y (i, j)

Algorithme de décodage BTC

A la réception nous avons A et B et le bloc binaire, et pour reconstruire le bloc de la source X il suffit de remplacer les éléments qui ont la valeur 0 par A , et les élément qui ont la valeur 1 par B.

$$X(i,j) = \begin{cases} A & Y(i,j) = 0 \\ B & Y(i,j) = 1 \end{cases} \tag{II-16}$$

II-4-3- Méthodes de compression par transformées

La compression par transformées est l'une des techniques de compression les plus utilisées, elle n'agissent pas directement sur la donnée dans sa représentation canonique, mais sur le domaine de sa transformée. Elle permet d'aboutir à un taux de compression très acceptable.

Le schéma de compression par transformée le plus célèbre est représenté sur la figure [II-12]:

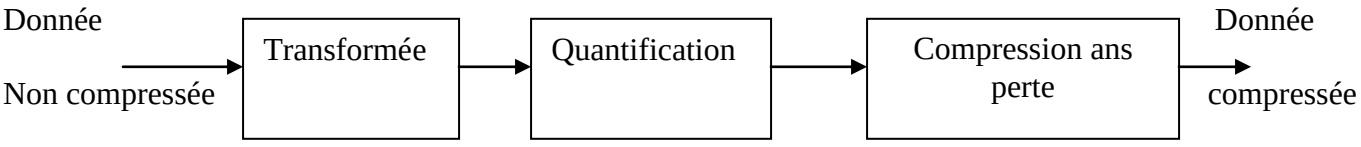


Figure II-12- Schéma d'un compresseur par transformée

1- Transformée

La première phase de compression dans le schéma général de compression, représenté sur la figure II-12, est la phase de transformation. Cette phase consiste principalement à appliquer une transformation mathématique aux données d'entrée. L'objectif global de cette transformée est de concentrer l'énergie initiale contenue dans les données initiales dans un nombre limité de coefficients. C'est-à-dire un nombre important de cœfficients ont des valeurs très faibles, qui peuvent être supprimés, ou se voir allouer un nombre limité de Bits pour les représenter.

Finalement la phase de transformée consiste à concentrer l'énergie de la source de données en peu de cœfficients :

Si E : est l'énergie d'un signal x (n), E est donnée par :

$$E = \sum_{n=1}^{N} x^2(n) \qquad \text{(II-17)}$$

E_T est l'énergie du signal x (n) dans le domaine de transformée, E_T est donc donnée par :

$$E_T = \sum_{n=1}^{N} T\big(x(n)\big)^2 \qquad \text{(II-18)}$$

2- Quantification

Cette phase a pour rôle également de réduire le nombre de Bits nécessaires pour représenter (coder) le signal transformé T(x(n)), avec une dégradation acceptable.

3- Compression sans perte

Cette phase consiste à appliquer un des compresseurs efficaces sans perte, ceci est afin d'assurer au maximum un taux de compression élevé, tels que : Huffman, arithmétique, RLE, BWT, ……

Il existe un autre schéma bloc de compression par transformée. La figure (II-13) présente les phases principales de compression :

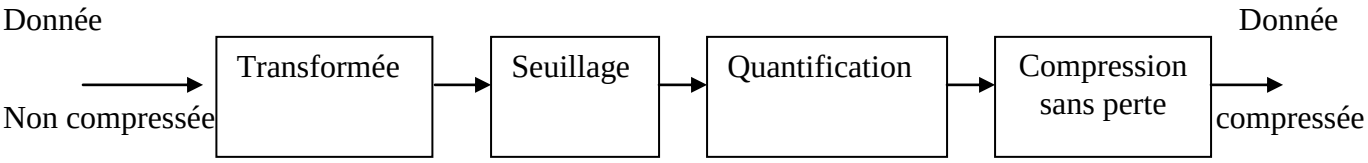


Figure II-13- Schéma efficace de compression par transformée

Pratiquement la seule différence est la phase de ***seuillage,*** ou cette phase consiste a supprimer (annuler) les coefficients qui ont des valeurs très faible, en fixant un seuil TH (threshold). Par conséquent, si T(x (n)) < TH alors T(x (n))=0.

Exemple :

Supposant la séquence de donnée suivante : le seuil S=5

T(x(n)) : 15 3 7 21 5 6 42 51 20 1 0 2 3 4

$T(x(n))_{seuillé}$= 15 0 21 0 6 42 51 20 0 0 0 0 0

Dans la section suivante on présente brièvement les transformées les plus connues, transformée de Hadamard, transformée en cosinus discrète (DCT), et la transformation par ondelette.

II-4-3-1- Transformée de Walsh Hadamard (WHT)

La transformée de Hadamard est une des transformées orthogonales de base. Une transformation qui est particulièrement simple pour être mise en application.

Les familles de fonctions de Walsh et Hadamard forment des systèmes de fonctions orthonormées, complets et dénombrables. Elles sont donc a priori également valables pour constituer des noyaux de transformations binaires.

Soit une suite périodique x (k) dont chacune des périodes contient N échantillons x(0), x(1), . . ., x(N - 1) ; sa transformée de Hadamard est, par définition, une autre suite

périodique Y(n) dont chacune des périodes contient également N échantillons Y (0), Y (1), .., Y (N - 1) définie par la relation :

$$Y_n = \sum_{k=0}^{N-1} x_k.(-1)^{(k,n)} \qquad \text{(II-19)}$$

Avec n, k = 0, 1, . . ., N - 1 (n, k étant exprimés en représentation binaire), N une puissance de 2 et (.) représente le produit scalaire binaire. Les transformées de Hadamard étant linéaires, nous pouvons adopter une notation matricielle

$$Y = H_N.x \qquad \text{(II-20)}$$

Y représente un vecteur du domaine transformé, x un vecteur du domaine temporel et H_N représente la matrice de Hadamard (N, N). Cette matrice possède les propriétés de symétrie ($(H_N)^t = (H_N)$), d'orthogonalité (équation II-21), de récurrence (équation II-22) et respecte l'égalité de Perceval. [23][21]

- Matrice de Hadamard

La matrice de Hadamard est une matrice carrée, où ces éléments sont soient des 1 soient des -1, rangés d'une façon que les colonnes sont orthogonaux. Si H est une matrice de Hadamard N x ***N*** le produit de H et sa transposée est la matrice identité.

$$H*H^T=NI \qquad \text{(II-21)}$$

I : la matrice identité

H : matrices de Hadamard dont les dimensions sont une puissance de deux peuvent être construites de la façon suivante :

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \qquad \text{(II-22)}$$

Avec H_1=[1], Par conséquent :

$$H_2 = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \text{(II-23)}$$

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad \text{(II-24)}$$

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad \text{(II-25)}$$

II-4-3-2- Transformation en cosinus discrète (DCT)

La DCT (Discrete Cosine Transform) est une transformation mathématique qui transforme une donnée présentée dans le domaine spatial en un spectre de fréquence, elle été très utilisée dans plusieurs domaines notamment, la compression de données. La DCT possède une excellente propriété de regroupement de l'énergie, afin de porter la majeure partie de l'information sur les coefficients qui ont une basse fréquence d'apparitions [16]

La DCT forme la base de beaucoup d'algorithmes de compression d'image et de la vidéo, particulièrement le JPEG standard de compression d'images fixes, et les normes vidéo MPEG-1 et , MPEG-2, et MPEG-4. Puisqu'une image est un signal bidimensionnel, la DCT bidimensionnelle est appropriée en termes de compression d'image. La DCT bidimensionnelle peut être calculée utilisant la DCT unidimensionnelle horizontalement et puis en utilisant le DCT 1-D verticalement.[11,31]

La DCT unidimensionnelle est définie par :

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\Pi(2x+1)u}{2N}\right] \quad , \quad \text{pour } x = 0, 1, \ldots..N\text{-}1. \quad \text{(II-26)}$$

Où

$$C(u)=\begin{cases}\frac{1}{\sqrt{2}} & ,u=0\\ 1 & ,ailleurs\end{cases} \qquad \text{(II-27)}$$

L'entrée est un ensemble de N valeurs f(x) (pixels, symbole audio, autre donnée), et la sortie est un ensemble de N coefficients de la transformée DCT C(u), le premier cœfficient C(0) s'appel DC (Direct component) et tous les autres coefficients sont appelés AC (Alternatives components).[10]

Donc la DCT d'un bloc de donnée (bloc de pixels dans une image) implique que les premiers coefficients de chaque bloc sont placés en haut à gauche du bloc correspondant. Plus on s'éloigne plus leurs grandeurs tendent à diminuer. Ce qui signifie que la DCT concentre la représentation de l'image en haut à gauche de la matrice de sortie [16]

L'équation ci-dessous représente la transformée inverse de DCT (IDCT) :

$$f(x)=\sqrt{\frac{2}{N}}\sum_{u=0}^{N-1}C(u)F(u)\cos\left[\frac{\Pi(2x+1)u}{2N}\right] \text{ Pour } x=0, 1, \ldots\ldots\ldots N-1 \qquad \text{(II-28)}$$

Dans le cas de l'image (2-dimentionnelle), on peut appliquer également la transformée en cosinus discrète en deux dimension (2D-DCT) sur un block de donnée M x N .

$$F(u,v)=\frac{2}{\sqrt{MN}}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}f(x,y)\cos\left[\frac{\Pi(2x+1)u}{2N}\right]\cos\left[\frac{\Pi(2y+1)v}{2M}\right] \qquad \text{(II-29)}$$

Pour $u = 0, 1, \ldots\ldots N-1$, et $v = 0, 1, \ldots\ldots\ldots M-1$

Où

$$C(k)=\begin{cases}\frac{1}{\sqrt{2}} & ,k=0\\ 1 & ,ailleurs\end{cases} \qquad \text{(II-30)}$$

Exemple

Soit la séquence de données suivante : f(x) = (12, 10, 8, 10, 12, 10, 8, 11), en appliquant la transformée DCT unidimensionnelle nous obtenons:

C(u) = (**28.6375**, 0.571202, 0.46194, 1.757, 3.18198, −1.72956, 0.191342, −0.308709).

L'application de IDCT donne la séquence suivante :

12.1883, 10.2315, 7.74931, 9.20863, 11.7876, 9.54549, 7.82865, 10.6557.

II-4-3-3- Transformation en ondelettes

Il est bien connu que la transformée de Fourier comme étant une généralisation de la série de Fourier –caractérisant les signaux périodique -. L'inconvénient majeur et quelle ignore complètement la contribution temporelle exacte d'une fréquence dans un signal.[8]

La transformée en ondelettes est une transformée mathématique, qui contrairement à la transformée de Fourier, permet d'obtenir des composantes localisées à la fois dans le domaine temporel et fréquentiel.

La transformée en ondelettes s'obtient ainsi (a et b étant des cœfficients de translation et dilation que l'on fait varier afin de couvrir le plan temps fréquence) par :

$$W(a,b) = \int_{-\infty}^{+\infty} X(t).\psi_{a,b}^{*}(t).dt \qquad \text{(II-29)}$$

Il est nécessaire de choisir, l'onde mère ψ (parmi les ondelettes). Il est possible de citer le chapeau mexicain, l'ondelette de Morlet ou l'ondelette de Haar). [17]

Algorithme de MALLAT

Cet algorithme est schématisé par la structure montrée dans la figure II.14:

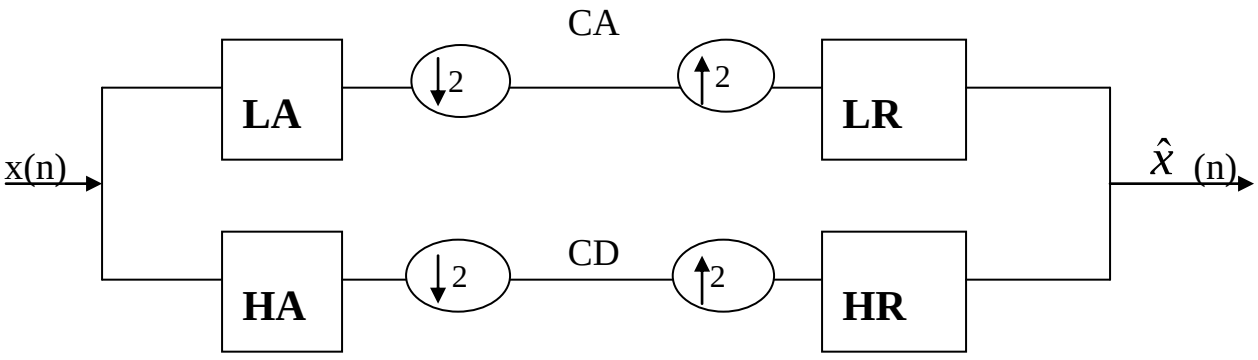


Figure II-14- Algorithme de MALLAT unidimensionnel

Le signal x (t) est décomposé par les deux filtres d'analyse LA (filtres basses fréquences) et HA (filtre hautes fréquences) pour donner deux vecteurs respectivement CA (coefficients ondelette d'approximation) et CD (coefficients ondelette de détails). Tous les deux sont de taille approximativement égale à la moitié du vecteur d'origine.

La reconstruction parfaite et théoriquement possible. Les deux vecteurs CA et CD en les faisant passer par un sur échantillonnage puis un filtrage respectivement par les filtres LR (filtre passe haut de reconstruction), et LH (filtre passe haut de reconstruction) donnent en somme le vecteur d'origine $\hat{x}(n)$.[8]

Le procédé de compression par ondelettes réalise une analyse par ondelettes en multi résolutions : on découpe l'image originale en quatre composantes avec des filtres passe-haut et passe-bas : l'une des composantes décrit l'allure générale de l'image, l'autres ses détails ; ces détails font l'objet d'une transformation par ondelettes et d'une quantification. Le procédé est ensuite répété pour une résolution demie de la précédente.[24]

Format JPEG2000

Nous décrivons ici les étapes du processus complexe de compression utilisé par JPEG2000 [43] :

1- L'image est convertie de la base RVB en base YCbCr.
2- Les composantes de chrominance Cb, Cr sont sous échantillonnées.
3- Pour chaque composante, l'image est découpée en régions (tuiles) : ce découpage est réalisé afin de réaliser des transformations en ondelettes indépendantes pour chacune de ces régions
4- (pour des raisons d'économie de mémoire). Le découpage en tuiles est cependant à éviter car pouvant occasionner des frontières visibles sur l'images compressée.
5- Une transformée en ondelettes est réalisée sur chaque tuile jusqu'à une profondeur arbitraire.
6- Une quantification est réalisée sur les coefficients obtenues par la transformée en ondelettes.

7- Les sous bandes quantifiées sont découpées en zones (precincts).

8- Les zones sont découpées elles-mêmes en blocs.

9- Chaque plan de bits d'un bloc est codé en trois passes en utilisant l'algorithme EBCOT.

10- Les bits sélectionnés par l'étape précédente font l'objet d'un codage arithmétique.

11- Le résultat du codage est organisé en paquets, eux-mêmes empaquetés dans des couches : les couches sont placées dans le flux dans un ordre adéquat afin d'assurer une augmentation

12- progressive de la qualité de l'image décodée lors du traitement de nouveaux paquets.

II-5- Conclusion

Depuis l'invention de l'informatique, on a toujours cherché à minimiser la place des données (texte, image, son, vidéo …) en mémoire. Aujourd'hui le volume d'information ne cesse de grandir avec la grande utilisation de l'Internet.

Dans ce chapitre nous avons présenté les principales méthodes de compression de données sans et avec perte. En terme de technologie comment arrive t-on à compresser des données La compression de données n'est pas purement un problème informatique. En effet, si Les recherches dans le domaine de la compression évoluent grâce à l'avancée de l'informatique, elles s'appuient surtout sur des théories mathématiques récentes. Le développement de ces méthodes s'avère aussi utile dans d'autres disciplines.

Il serait enfin prématuré de prédire la disparition de la compression de données au profit de la performance des machines. Car si la capacité de nos ordinateurs grandit sans cesse, la taille des fichiers à conserver croit de manière identique. Il faut donc s'attendre à voir apparaître des méthodes de compression toujours plus rapide et plus efficaces

Transformée de Burrows Wheeler

BWT

III-1- Introduction

Dans ce chapitre nous allons présenter la transformée de Burrows-Wheeler (BWT) qui constitue, en réalité, la partie centrale de notre travail. Quoi qu'elle soit incluse dans une chaîne de compression, elle est considérée comme une phase de prétraitement pour transformer les données initiales en une forme plus adaptée à la compression. La caractéristique essentielle de celle-ci se manifeste en son pouvoir de réarranger les codewords similaires en des séries de successions le plus possible.

III-2- Caractéristiques de la BWT

Il est clair que la meilleure façon d'ordonner un ensemble de données est de les faire trier « Sorting » Soit d'une manière ascendante ou descendante. Cependant pour revenir aux données initiales et gagner en compression, ceci est pratiquement impossible, d'où est venue l'idée de la transformée de Burrows-Wheeler.

Ses caractéristiques principales sont :

- C'est une technique d'arrangement par bloc« block-sorting ». [20]
- Elle est réversible. Mathématiquement dit c'est une fonction bijective [26]
- Aucune réduction de taille après l'application de la BWT, mais ou contraire, elle ajoute à chaque bloc transformé un index.
- Sa vitesse d'exécution diminue significativement en fonction de la taille du bloc.

III-3- Principe d'un schéma de compression par BWT

Un schéma usuel de compression utilisant la BWT est donné par la figure (III-1) [20] :

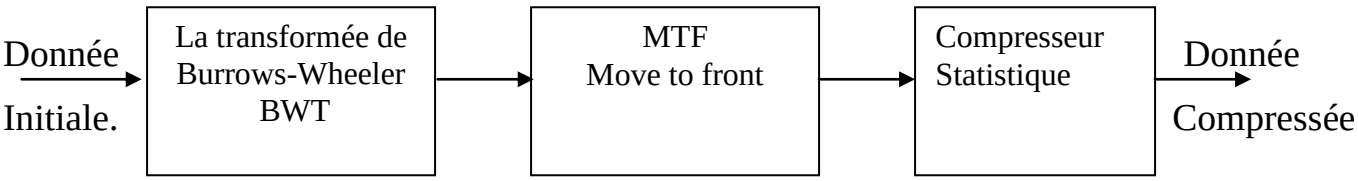


Figure III-1- Schéma usuel de compression utilisant la BWT

- La partie centrale de ce schéma est la BWT qui permute et réarrange la donnée en une forme adaptée aux deux étapes qui suivent.
- la transformée Move To Front (MTF) qui consiste à un remplacement d'une succession d'un même code

$$\overbrace{xxxx\ldots\ldots\ldots\ldots x}^{N} \quad \text{Par} \quad \overbrace{indx0000\ldots\ldots\ldots 0}^{N}$$

ou *indx* est l'index actuel du codewords *x* dans la table d'index de la MTF.

- Finalement un compresseur statistique donnera un résultat de compression efficace.

III-4- Transformées directe BWT [25]

Une description détaillée de la transformée directe BWT est donnée en ce qui suit :

Prenons l'exemple de la séquence d'entrée « a a r d v a r k $», elle est issue de l'alphabet «$, a, d, r, k, v », l'algorithme de la BWT est le suivant :

- ***Etape1*** : si la taille de la séquence est N (9 dans notre exemple), nous devons effectuer N rotations y compris la séquence d'origine (figure III-2)

a	a	r	d	v	a	r	k	$
a	r	d	v	a	r	k	$	a
r	d	v	a	r	k	$	a	a
d	v	a	r	k	$	a	a	r
v	a	r	k	$	a	a	r	d
a	r	k	$	a	a	r	d	v
r	k	$	a	a	r	d	v	a
k	$	a	a	r	d	v	a	r
$	a	a	r	d	v	a	r	k

Figure III-2 : Matrice issue des rotations

Une matrice carrée N*N est produite.

- ***Etape 2*** : cette étape consiste à faire ordonner « Sorting », les lignes de la matrice résultante d'une manière lexicale. Dans notre exemple le $ est supposé ayant le codage le plus petit. La figure (III-3) représente la nouvelle matrice qui en résulte

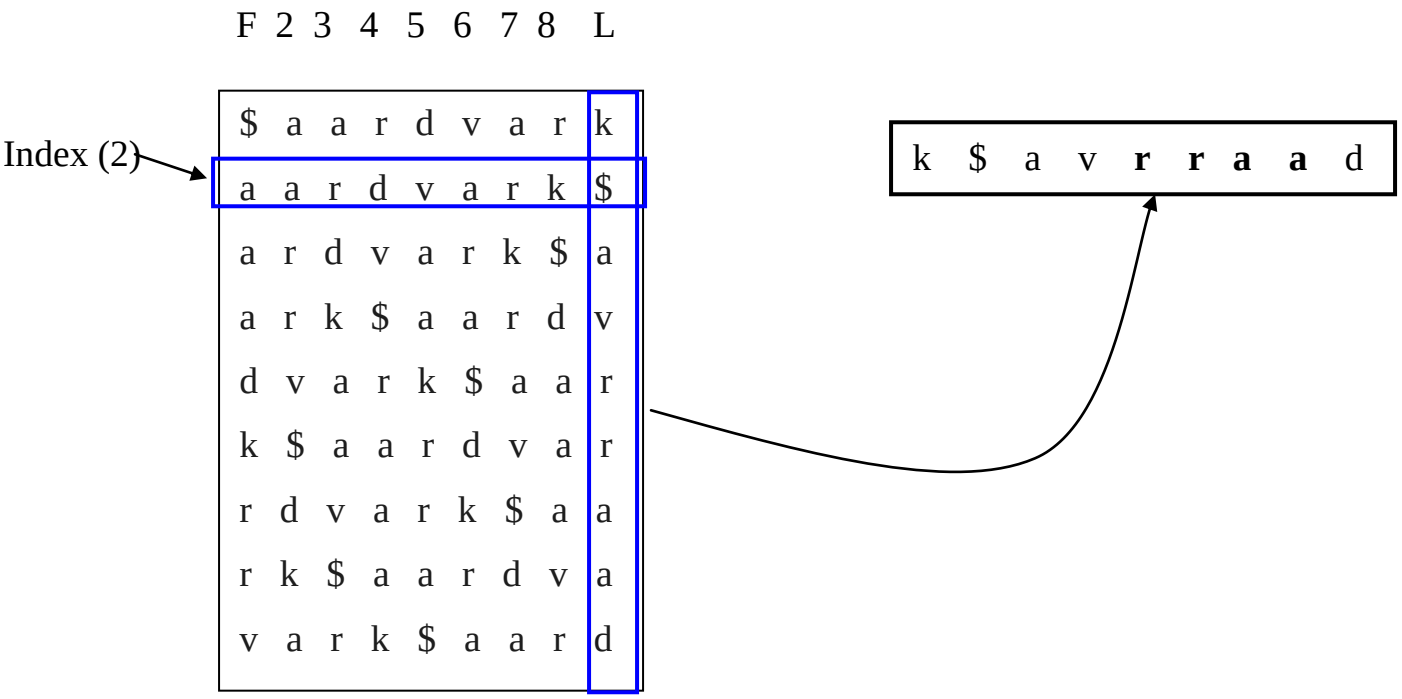


Figure III-3- Matrice réarrangée lexicalement

- ***Etape 3***: la dernière colonne de la dernière matrice représente la séquence de sortie de la BWT. Ainsi, dans notre exemple, la séquence est : « k $ a v r r a a d ». Ici, nous remarquons deux successions qui apparaissent (a a) et (r r) ;
- ***Etape 4*** : sauvegarde de l'index de la séquence initiale (2). Il marque la position de cette séquence dans la matrice réarrangée lexicalement.

III-5- Transformée inverse (IBWT) [25]

La transformée inverse est, comme précédemment évoqué, se base sur l'index, la séquence de sortie de la BWT et finalement sur la séquence de sortie ordonnée lexicalement (dans notre exemple « $ a a a a d k r r v »).

En réalité, il s'agit d'utiliser trois paramètres pour retrouver la matrice réarrangée lexicalement et par la suite la séquence initiale.

La matrice réarrangée est issue à partir de la manipulation de la séquence de sortie et la séquence ordonnée

Continuions avec notre exemple, en premier lieu deux colonnes de la matrice sont données (figure III- 4) :

F	2	3	4	5	6	7	8	L
$								k
a								$
a								a
a								v
d								r
k								r
r								a
r								a
v								d

Figure III-4 : La première et la dernière colonne de la matrice réarrangée

Nous insérons une copie de la dernière colonne avant la première. Par la suite nous faisons un tri lexical ligne par ligne des nouvelles séquences constituées, chacune de deux codewords. (Figure III- 5)

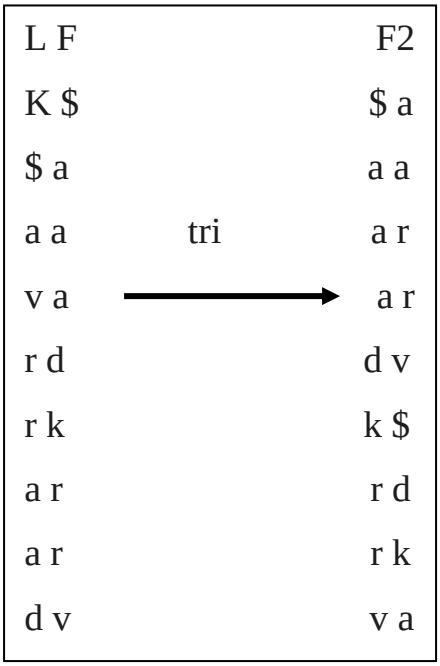


Figure III-5 : Les deux premières colonnes générées

Insérons, de nouveau la dernière colonne, comme précédemment effectué, pour obtenir la figure (III – 6)

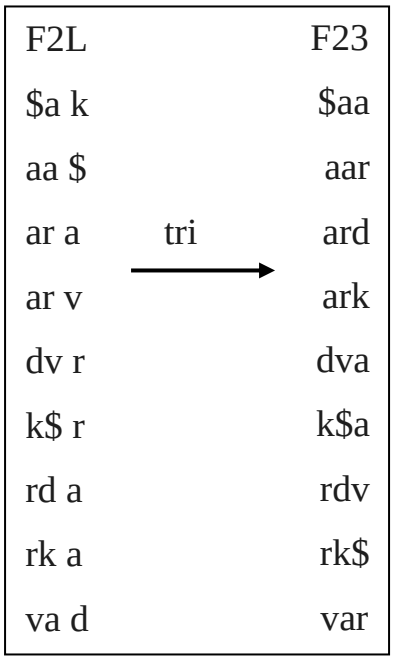


Figure III - 6 : Les trois premières colonnes générées

Dans la même optique et en suivant le même raisonnement, nous obtenons à la fin une matrice carrée réarrangée lexicalement. Ainsi, nous utilisons l'index sauvegardé pour extraire la ligne qui représente la séquence initiale.

Il est à noter que la transformée inverse est gourmande en temps d'exécution. Ainsi, des travaux de recherche sont en cours pour réduire le temps d'exécution [27,28]

III-6- Transformée move-to-front (MTF)

Pour clarifier l'impact de l'utilisation de la transformée « MOVE-TO-FRONT », nous donnons un exemple (Figure III.7)

Séquence d'origine :

230 228 261 257 230 228 261 257 230 228 261 257 230 228 261 257 230 228 261 257 230 228 261 257

Séquence après BWT

230 230 230 230 230 230 257 257 257 257 257 257 261 261 261 261 261 261 228 228 228 228 228 228

Figure III-7 : Exemple pratique de la BWT

Comme nous pouvons remarquer la séquence de sortie de la BWT montre plusieurs successions de codewords.

La phase qui suit logiquement est la MTF qui consiste au remplacement de chaque succession par un index et une suite de zéros

Son principe est le suivant :

Soit une séquence d'entrée de la figure III-7 issue de l'alphabet " 228, 230, 257, 261", l'algorithme MTF consiste à remplacer chaque codewords de la séquence d'entrée par son index qui représente sa position dans un tableau qui contient l'ensemble des symboles de l'alphabet, ce dernier sera modifié en mettent le symbole codé en tête, et ainsi de suite,

jusqu'au dernier symbole à codé. Le tableau III-1 représente le codage MTF du premier jusqu'au 8eme symbole.

Tableau III-1 : Exemple pratique de l'algorithme MTF pour les 8 premiers symboles

	0	1	2	3
	228	230	257	261
1	230	228	257	261
0	230	228	257	261
0	230	228	257	261
0	230	228	257	261
0	230	228	257	261
0	230	228	257	261
2	257	230	228	261
0 . .	257	230	228	261

À la fin nous obtenons une séquence de codes MTF, la figure III-8 représente la séquence complète du codage MTF.

Séquence après BWT

230 230 230 230 230 230 257 257 257 257 257 257 261 261 261 261 261 261 228 228 228 228 228 228

Séquence après MTF :

2 0 0 0 0 0 3 0 0 0 0 0 4 0 0 0 0 0 3 0 0 0 0 0

Figure III-8 : Codage MTF complet de la séquence

Il est clair que la MTF rend la donnée plus compressible en utilisant les compresseurs statistiques (exemple : HUFFMAN), puis elle fais apparaître le 0 comme caractère fréquent. Cependant, le codage par RLE aussi peut produire un bon résultat.

III-7- Conclusion

Nous avons présenté dans ce chapitre, la transformée BWT et nous avons montré ses caractéristiques essentielles. Associée à la MTF et à un compresseur sans perte, elle peut se qualifier comme une étape de prétraitement indispensable menant à une compression efficace.

Applications

IV-1- Introduction

La compression de données (data compression), consiste à représenter la donnée sous une forme qui occupe moins d'espace mémoire lors de stockage, ou de réduire le temps de transmission à travers un système de communication. Ceci est assuré d'une manière générale en deux étapes: dans le premier lieu, on cherche à représenter les données sous une forme plus adaptée à la compression. Dans le deuxième lieu, on code ces données, ce qui nous permet de conserver l'information importante.

L'objectif de notre travail est de chercher à effectuer efficacement la première étape de changement de représentation pour les données textes et images à niveaux de gris. Nous nous sommes placés dans un cadre de changement de représentation par la transformée de Burrows Wheeler (BWT), cette dernière est considérée comme une phase de prétraitement de données afin d'augmenter la compressibilité par un réarrangement de données (voir chapitre III).

IV-2 - Compression texte

Un schéma typique d'un algorithme de Compression fondé sur la transformée de Burrows Wheeler (BWT) est montré sur la figure (IV-1). Il est constitué de trois étapes.

L'étape 1 est la BWT qui effectue un arrangement le plus possible. Ces données de sorties sont prétraitées pour la deuxième étape.

L'étape 2 est celle de l'utilisation de la MTF qui remplace les successions de différents codes par des indexes et de suites de zéros. Il est évident, qu'une source de données qui présente des successions fréquentes de codes est transformée, par la MTF, en une autre où le zéro est le code dominant.

L'étape 3 utilise un encodeur sans perte de type Huffman, arithmétique…[29]

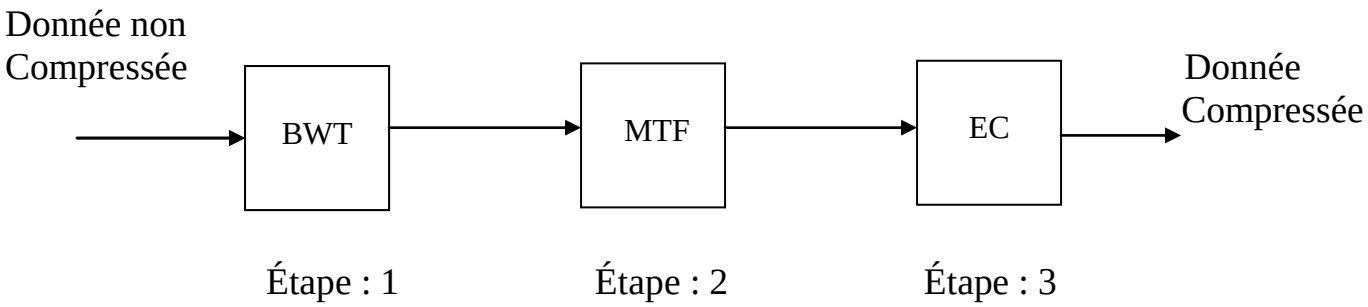


Figure IV-1 : Structure typique de la compression par BWT

La figure IV. 2. montre les différents résultats des différentes étapes appliquées sur le texte original « mississippi mississippi ».

	m	i	s	s	i	s	s	i	p	p	i	m	i	s	s	i	s	s	i	p	p	i
(a) Entrée BWT:	6D	69	73	73	69	73	73	69	70	70	69	6D	69	73	73	69	73	73	69	70	70	69
(b) Sortie BWT :	70	70	73	73	73	73	6D	6D	69	69	70	70	69	69	73	73	73	73	69	69	69	69
(c) Sortie MTF :	70	00	73	00	00	00	6F	00	6C	00	03	00	01	00	03	00	00	00	01	00	00	00

Figure IV-2: Compression de la séquence « mississippimississippi »

IV-3- Résultats et analyse

Il est clair qu'il y a un problème de choix de taille du bloc à traiter par la BWT car il n'y a pas de règle mathématique pour déterminer la taille optimale du bloc. Pour cela nous avons appliqué notre programme de compression aux premiers 40 Kilo Octets (pour différentes tailles de segments à traiter par la BWT de: 1, 2 et 4 Kilo Octets) pour une série de fichiers de test (benchmark text files [33]). Les résultats de compression sont représentés dans le tableau IV-1, à la première colonne nous avons les noms des fichiers, dans la deuxième colonne les taux de compression pour un segment BWT de 1 kilo octets et dans la troisième et la quatrième colonne, nous avons présenté les taux de compression pour les segments BWT de tailles 2 et 4 kilo octets respectivement.

Tableau IV-1 Résultats de compression pour les déférentes tailles de segments à traiter par la BWT

Nom des fichiers	Taux de compression (BWT)		
	Seg =1ko	Seg = 2 ko	Seg = 4 ko
Bib	1.90	2.07	2.26
Book1	1.90	2.00	2.11
Book2	1.95	2.09	2.24
Paper1	1.93	2.09	2.26
Paper2	1.95	2.08	2.21
Trans	2.10	2.30	2.55

Le tableau IV- 1- confirme clairement que la taille 4ko (4096B), est la taille optimale par rapport aux deux autres tailles. A partir de ces résultats, nous pouvons dire que plus que la taille du bloc est grande plus que la BWT donne des bons résultats, mais nous nous somme limité à cause de la puissance de l'ordinateur d'exécution à la taille de 4096 octets.

Pour vérifier l'efficacité du programme élaboré utilisant la transformée de Burrows-Wheeler, nous avons comparé nos résultats obtenus avec les résultats de compresseur WINRAR 3.7 (option: compression normale).

Tableau IV-2- Résultats de compression de textes par BWT

Nom de fichier	Taille Originale (octets)	Taille compressée (BWT)	CR(bwt)	Taille compressée (octets) (WINRAR 3.7)	CR WINRAR
Paper 1	53161	23014	2.3099:1	18159	2.9275
Paper 2	82199	36447	2.2553:1	28748	2.8593
Paper 3	46526	21290	2.1854:1	17824	2.6103
Paper 4	13286	6034	2.2017:1	5580	2.3810
Paper 5	11954	5385	2.2198:1	5034	2.3747
Paper 6	38105	16415	2.3214:1	13140	2.8999
Obj 1	21504	11593	1.8550:1	9800	2.1943
Obj 2	246814	100420	2.4578:1	71728	3.4410
Pic	513216	69750	7.3580:1	49677	10.3311
Book 1	768771	365800	2.1016:1	276962	2.7757
Book 2	610856	266720	2.2902:1	181075	3.3735
Geo	102400	72562	1.4112:1	62903	1.6279
Progp	49379	15768	3.1317:1	10834	4.5578
Bib	111261	49255	2.2589:1	33049	3.3665
Progl	71646	22477	3.1876:1	15799	4.5348
Progc	39611	16478	2.4039:1	13199	3.0011
News	377109	183560	2.0544:1	125999	2.9930
Trans	93695	34509	2.7151:1	18033	5.1958

Les résultats obtenus nous indiquent que la BWT est adaptées à ce type de fichier. Ainsi, l'algorithme de compression appliqué est proche en performance de WINRAR pour la majeure partie des 18 fichiers de test présentés dans le tableau IV-2. Cependant, plusieurs optimisations pourraient être apportées à l'algorithme de compression afin de la rendre plus efficace.

IV-4- Compression des images niveaux de gris

IV-4-1- Image niveaux de gris

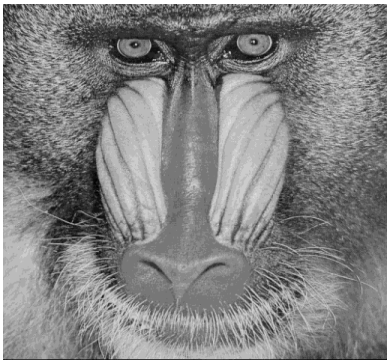
En général, les images en niveaux de gris renferment 256 couleurs de gris. C'est une image où chaque pixel à un codage d'intensité lumineuse de 8 bits. Par convention la valeur zéro indique le noir (intensité lumineuse la plus sombre) et la valeur 255 le blanc (intensité lumineuse la plus éclairée).[31]

La figure IV. 3 donne un ensemble d'images de test les plus utilisées pour le traitement d'image (exceptée l'image Sarah).[32]

Lena

peppers

Barbara



Baboon

Goldhill

image1

image2

Sarah

Figure IV-3 : Ensemble d'images niveaux de gris

IV-4-2- Compression des images par BWT

En ce qui suit nous présentons la structure proposée :

IV-4-2-1- Structure de la chaîne de compression des images

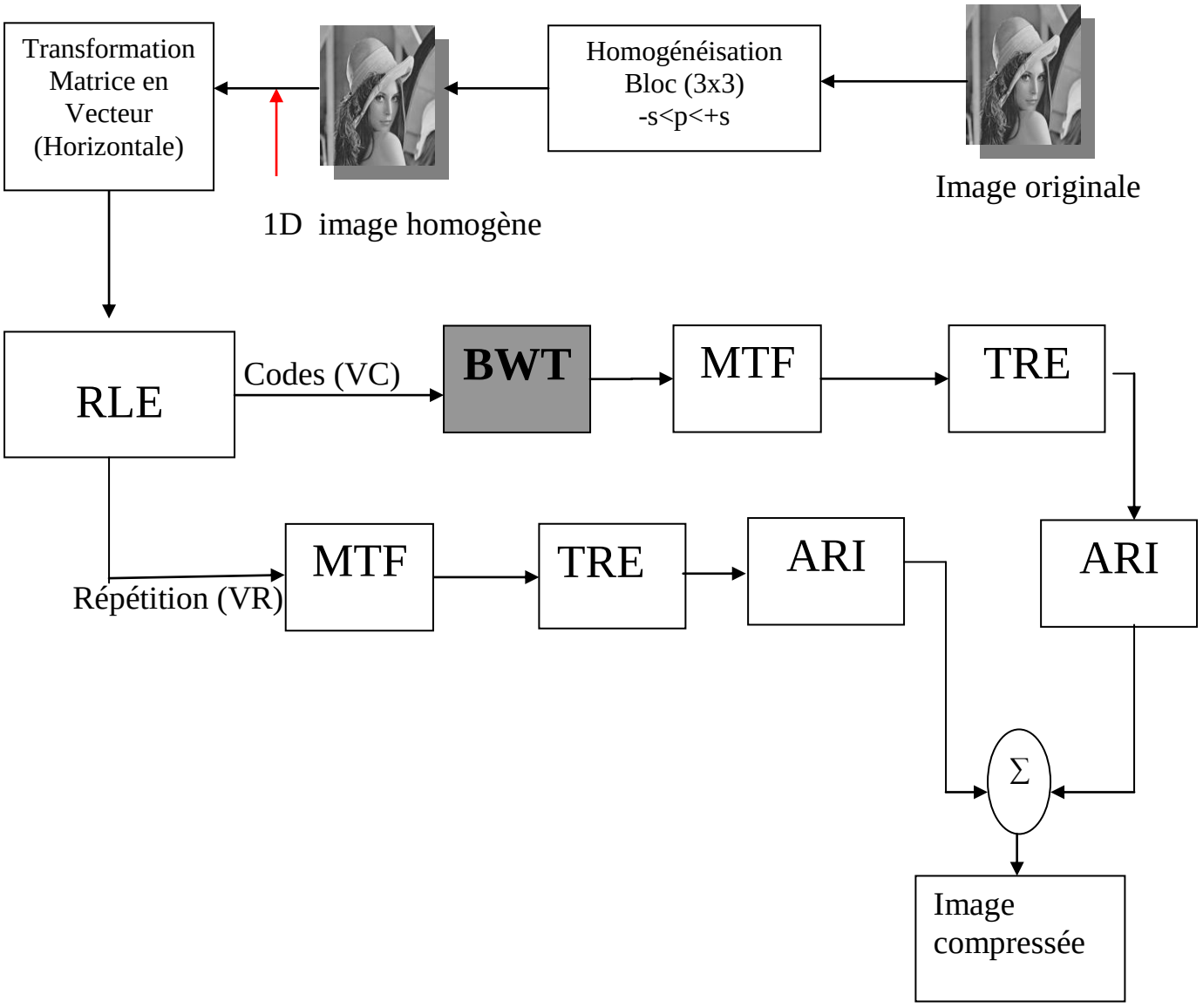


Figure IV-4: Chaîne de compression des images incluant la BWT

IV-4-2-1-1- Homogénéisation des blocs (3*3) -s<p<+s

Ce bloc a pour rôle de faire un seuillage autour du pixel central de chaque bloc 3*3 de l'image originale. Ceci, afin d'augmenter la redondance dans l'image, tout en gardant la qualité de résolution vérifiée en tenant compte du PSNR pris comme mesure de qualité.

Exemple : La figure IV-5 représente les 6 premiers blocs de l'image Lena avec s=2

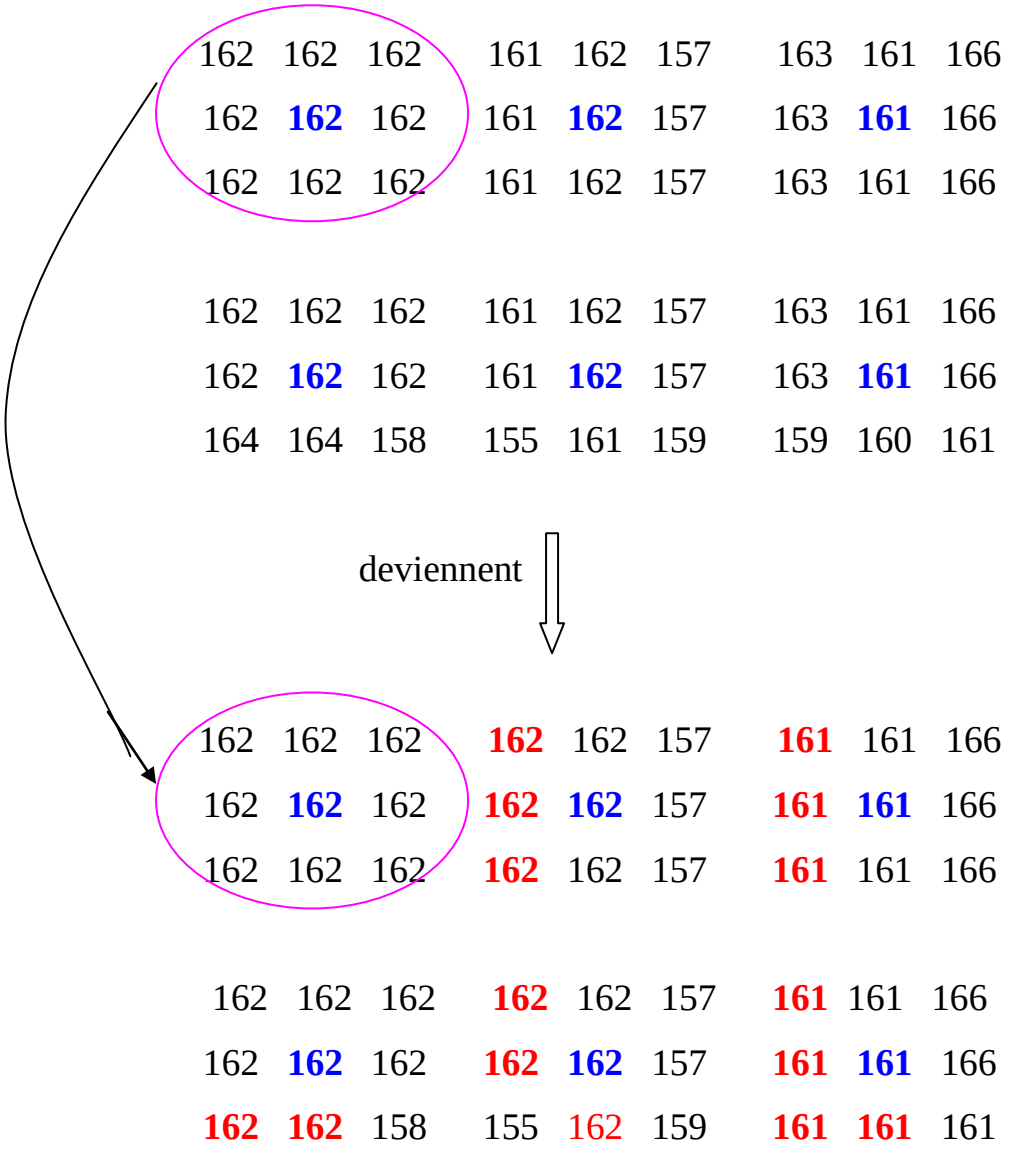


Figure IV-5: Les 6 premiers blocs de l'image Lena avec s=2

La figure IV.6 montre le résultat de cette étape pour les images « Lena » et « Sarah ».

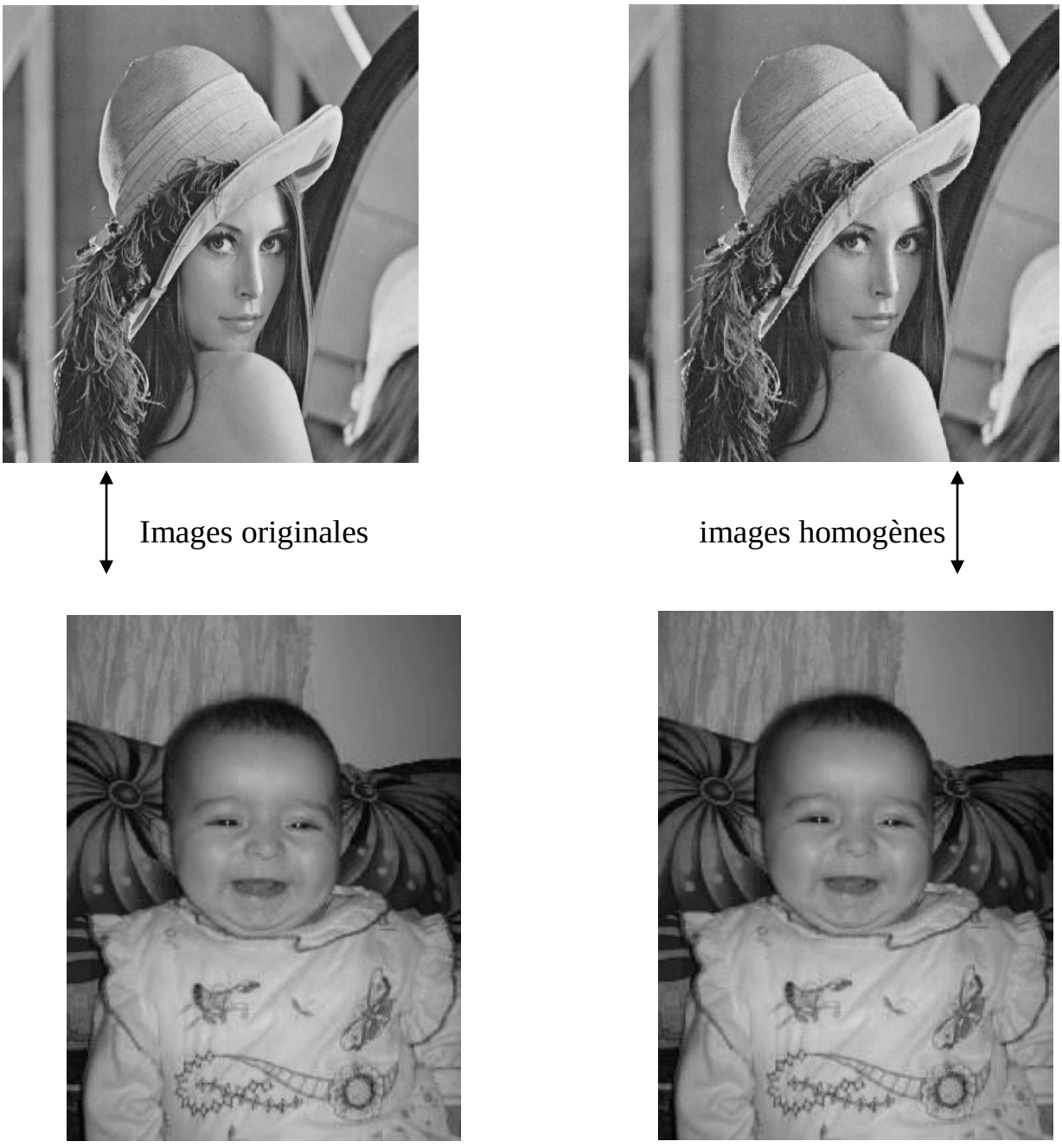


Figure IV-6: Exemple d'i mages homogénéisées

IV-4-2-1-2- Transformation image-vecteur

L'image seuillée (taille NxM) est transformée (par un balayage horizontale ligne par ligne) en un vecteur de tail1e NxM (figure IV-7-a), mais il peut y avoir d'autres manières de balayage : Verticalement (figure IV-7-b), en zigzag (figure IV-7-c)…

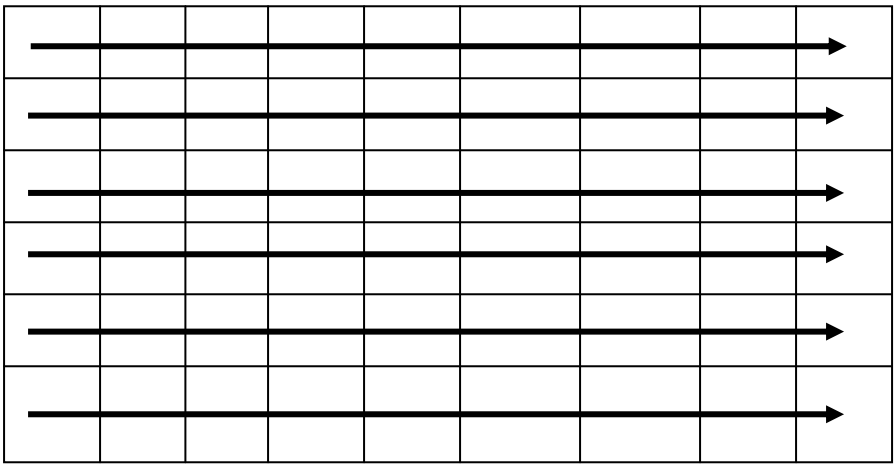


Figure IV- 7- a : Balayage horizontal

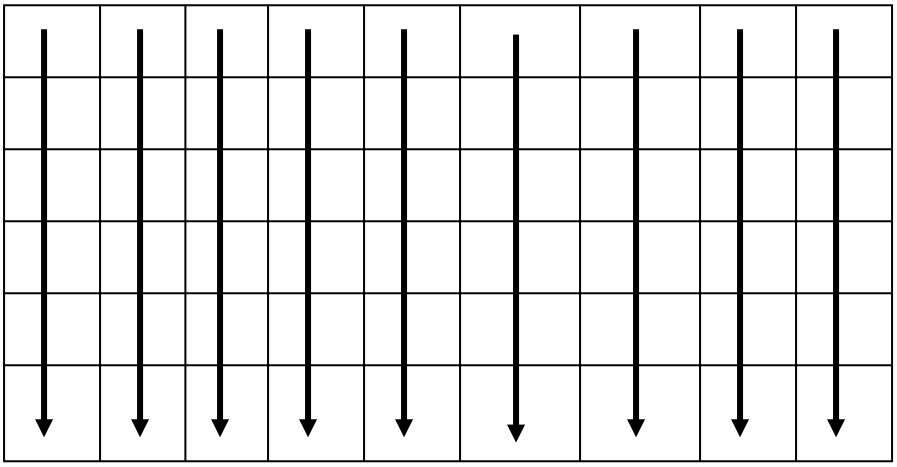


Figure IV- 7- b : Balayage vertical

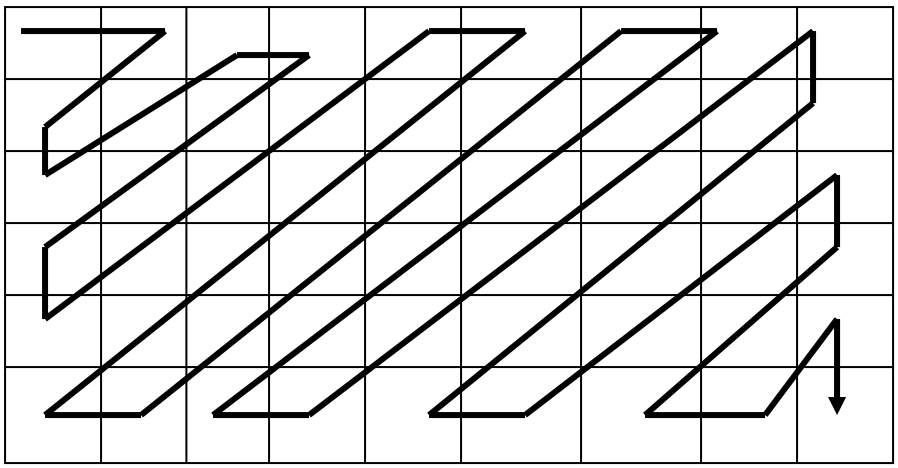


Figure IV- 7- c : Balayage en zigzag

IV-4-2-1-3- Codage RLE

Notre objectif d'introduire le codeur RLE ici a pour rôle de préparer le vecteur des codewords a se réorganiser par la transformée de burrows wheeler (BWT). Il est clair que la BWT regroupe les pixels du même niveau de gris qui sont espacés par contre si les pixels sont regroupés (cas des images), alors la BWT donc jouera le rôle inverse.

Après application de l'RLE nous obtenons deux vecteurs : VC qui est le premier vecteur représentant les codes et le deuxième vecteur VR qui est celui des valeurs de répétitions.

IV-4-2-1-4 BWT

Nous avons remarqué que le vecteur VC présente des codes dispersés qui est le cas favorable pour l'application de la BWT.

Nous tenons à signaler qu'avant l'application de la BWT, nous avons segmenté le vecteur VC en des segments de taille 4096. (A notre connaissance), il n' y a pas de règle pour déterminer la longueur optimale qui donne un temps de calcul raisonnable et un rendement d'homogénéisation acceptable.

Exemple : La figure IV-8 représente les 16 premiers pixels de lena:

Lena originale :	162	162	162	161	162	157	163	161	166	162	162	160	155	163	160	155
Lena homogène. :	162	162	162	**162**	162	157	**161**	161	166	162	162	**162**	155	163	160	**157**
RLE(VC) :	162	157	161	166	162	155	163	160	157							
RLE(VR) :	5	1	2	1	3	1	1	1	1							
BWT(VC) :	**162**	**162**	160	163	157	166	157	155	161							

Figure IV-8: Exemple de codage des 16 premiers pixels de lena:

IV-4-2-1-5- MTF

L'application de la MTF a pour objectif de remplacer les séquences répétées (redondance) par la valeur 0 afin d'augmenter la compressibilité de TRE.

Exemple :

BWT : 162 162 162 162 162 155 155 155 155 97 97 97 97 97 97

MTF : 162 0 0 0 0 0 156 0 0 0 99 0 0 0 0

IV-4-2-1-6- TRE [30]

Le codage TRE (Two-Role Encoder) se résume comme suit :

Supposons une suite de codes C1, C2,…..CN qui contient des successions de zéros (cas de la sortie de l'MTF) :

- Si un code Ci est différent de zéros on lui ajoute la valeur 256 et le nouveau code est codé sur 9 bits (on suppose ici que les codes Ci sont codés sur 8 bits chacun).
- Une succession de zéros >=1 et <256 est remplacée par le nombre de succession codé sur 9 bits.

IV-4-2-2- Résultats et analyse

Nous avons remarqué que l'application directe de l'algorithme d'origine [26] aux données image ne donne pas des résultats satisfaisants. Cependant, l'utilisation (le cas de notre travail) de l'RLE avant la BWT et l'application de cette dernière sur le vecteur VC améliore le rendement de la BWT. Pour pouvoir quantifier les résultats obtenus du schéma proposé, nous avons travaillé sur une série d'images de 512x512 octets. Les résultats sont présentés dans les tableaux ci-dessous, où dans la première colonne nous donnons les noms des images de test, dans la deuxième colonne, les tailles compressées par BWT, ensuite, dans la troisième colonne, le taux de compression et enfin à la dernière colonne nous donnons le rapport signal sur bruit (PSNR), ceci est pour les déférents seuils d'homogénéisation :

Tableau IV-3 : Résultats de compression d'image par BWT pour S=7

Nom d'image	Taille compressée BWT (octet)	Taux de compression CR	PSNR(dB)
Lena	89904	2.65:1	39.08
Barbara	137000	1.91:1	40.06
Baboon	194810	1.35:1	40.74
Peppers	100940	2.60:1	38.74
Image 1	34381	7.62:1	40.40
Image 2	140800	1.86:1	39.16
Goldhill	131650	1.99:1	39.20

Tableau IV-4 : Résultats de compression d'image par BWT pour S=10

Nom d'image	Taille compressée BWT (octet)	Taux de compression CR	PSNR(dB)
Lena	76744	3.4158	36.7767
Barbara	120660	2.1726	37.6089
Baboon	175420	1.4944	37.0602
Peppers	75919	3.4529	36.2276
Image 1	26611	9.8510	39.2976
Image 2	111360	2.3540	35.7162
Goldhill	104630	2.5055	36.1795

Tableau IV-4 : Résultats de compression d'image par BWT pour S=13

Nom d'image	Taille compressée BWT (octet)	Taux de compression CR	PSNR(dB)
Lena	63943	4.10:1	35.31
Barbara	109250	2.39:1	35.80
Baboon	158111	1.66:1	34.52
Peppers	61201	4.28:1	34.74
Image 1	23423	11.19:1	38.65
Image 2	88835	2.95:1	33.49
Goldhill	85179	3.08:1	34.19

Les résultats obtenus sont considérés satisfaisants, si nous prenons, par exemple l'image Lena, celle-ci est compressée avec un taux de compression de 2.65:1 et un PSNR de 39.08dB pour un seuil d'homogénéisation égal à 7, un taux de 3.41:1 avec un PSNR de 36.77 dB pour un seuil d'homogénéisation égal à 10, et enfin, un taux de 4.10:1 avec un PSNR de 35.31 dB pour un seuil d'homogénéisation égal à 13. Par contre l'image " Baboon" est compressée avec (taux, PSNR, seuil) de (1.34:1, 40.74, 7) et (1.49:1, 37.06, 10) et (1.66:1, 34.52, 13), les taux de compression pour cette image sont faible par rapport aux précédents parce que l'image Baboon contient beaucoup plus de détails.

Finalement, nous pouvons conclure que si nous augmentons la valeur du seuil S, l'augmentation du taux de compression est évidente, mais en conséquence une diminution de facteur de qualité (PSNR) sera remarquée. La figure IV-9- nous présente l'effet du seuil d'homogénéisation S sur la visibilité de l'image Lena et leur Zoome avec les déférents taux de compression et les différentes valeurs du PSNR.

Lena Originale

Lena: CR=2.65 PSNR=39.08dB avec S=7

Figure IV-9- Lena avec les différents seuils d'homogénéisation

Lena : CR=3.42 - PSNR=36.78dB avec S=10

Lena : CR=4.10 PSNR=35.31dB avec S=13

Figure IV-9- Lena avec les différents seuils d'homogénéisation

IV-5 Conclusion

Nous avons appliqué l'algorithme original incluant la BWT (cas du texte) et un algorithme proposé (en cas de l'image). Dans le cas de la compression de données textes, nous avons obtenus des résultats satisfaisants confirmant l'adaptabilité de la BWT à ce type. Cependant, pour le cas des images, nous pouvons dire que les résultats sont acceptables (Avec une bonne qualité), mais des améliorations, qui reste à faire, sont indispensables pour obtenir des résultats bien encore meilleurs.

Conclusion générale

La BWT, technique encore toute jeune, est considérée comme une phase de prétraitement dans une chaîne de compression de données [26]. Durant l'élaboration de notre travail, nous avons remarqué que la BWT agisse efficacement dans le cas de données initialement dispersées (cas du texte), jouant ainsi le rôle d'un algorithme de tri. Par contre, dans le cas de données initialement quasi-regroupées (cas d'une image), l'application directe de la BWT, ne donne pas généralement un résultat satisfaisant.

Pour la compression de texte, en appliquant directement la BWT au vecteur de données texte suivi par l'algorithme MTF puis ARI, les résultas obtenus sont satisfaisants.

Dans le cas de l'image la chose la plus importante est d'appliquer l'algorithme RLE avant la BWT pour assurer la dispersion et pour préparer le vecteur de données à la BWT afin de rendre son usage plus efface.

Egalement, nous avons vu et vérifié, grâce à notre application, qu'il est possible de combiner plusieurs algorithmes de compression pour donne, de meilleurs résultats que lorsqu'un seul algorithme est utilisé.

Notre modeste travail élaboré, est loin d'être parfait, mais lors de son élaboration nous avons constaté que des améliorations peuvent êtres apportés, à savoir :

- Travailler sur l'accélération de la IBWT pour obtenir, similairement à la FFT, la Fast IBWT ; [27,28]
- Changer, le domaine d'application et essayer de travailler sur les données génomiques [25];
- Explorer les autres applications de la BWT, exemple : la recherche d'un motif « Pattern Search ».[25]

Bibliographie

[1] Robert.B.Ash, « information theory » Urbana, Illinois, July, 1995

[2] Thomas M. Cover, Joy A. Thomas « Elements of Information Theory » John Wiley & Sons, Inc, New York, 1991

[3] Robert G. Gallager « information theory and reliable communication » John Wiley & Sons, Inc., New York, 1968

[4] Jacques CLAVIER, « théorie de l'information- Aspects mathématiques » techniques de l'ingénieur, traité électronique, E 3082-1:

[5] Robert M. Gray « Entropy and Information Theory » Electrical Engineering Department Stanford University, Springer-Verlag, New York, 1990.

[6] Brian C. Vickery Alina « Vickery Information Science in Theory and Practice » Third edition, K. G. Saur verlag GmbH, München, Germany, 2004

[7] Vincent Beaudoin, « Développement de nouvelles techniques de compression de données sans perte » mémoire de maître ès sciences (M.Sc), Université Laval Québec, 2008.

[8] Redha Benzid, « Ondelette et statistiques d'ordre supérieur Appliquées aux signaux Uni et bidimensionnels », Thèse de doctorat, université de Batna, 2005

[9] khalid sayood, « Lossless Compression Handbook » academic press series in communications, networking, and multimedia, Elsevier Science, New York, 2003

[10] David Salomon, « A Concise Introduction to Data Compression » Springer-Verlag London Limited, 2008.

[11] Tinku Acharya, Ping-Sing Tsai, « JPEG2000 Standard for Image Compression Concepts, Algorithms and VLSI Architectures » John Wiley & Sons, Inc, New Jersey, 2005,

[12] David Salomon, « Data compression » the complete reference 3^{e} edition, Editions Springer, 2004.

[13] David Salomon, « Variable-length Codes for Data Compression » Springer-Verlag

London Limited 2007.

[14] Atef MASMOUDI Mohamed Salim BOUHLEL, « Un nouvel algorithme de compression exploitant le codage arithmétique en lui introduisant de nouveaux paramètres de codage », Unité de Recherche: Sciences et Technologies, Institut Supérieure de Biotechnologie de Sfax (ISBS)-TUNIS, 2007.

[15] Roberto Togneri, Christopher J.S. deSilva, « fundamentals of information theory and coding design », A CRC Press Company, London, 2006.

[16] Darrel Hankerson, Greg A. Harris, Peter D. Johnson, Jr « Introduction to Information Theory and Data Compression », Second Edition, CRC Press LLC, New York 2003.

[17] Peter Wayner, « Data Compression for Real Programmers » Morgan Kaufman, Inc. USA, 1999.

[18] Thomas M. Cover, Joy A. Thomas « Elements of Information Theory » Second Edition, John Wiley & Sons, Inc, Canada, 2006.

[19] Mark Nelson, « the data compression book » Second Edition, M & T Books, IDG Books Worldwide, Inc, Cambridge, 2007.

[20] Khalid sayood, « introduction to Data compression » 3ed edition, Elsevier Inc., San Francisco, 2005.

[21] David Dagan Feng, « biomedical information technology » Elsevier Inc, Amsterdam, 2008.

[22] Meftah M. Almrabet , Amer R. Zerek, Allaoua Chaoui and Ali A. Akash, « Image Compression Using Block Truncation Coding » International Journal of Sciences and Techniques of Automatic control & computer engineering IJ-STA, Volume 3, N° 2, pp. 1046−1053, December 2009.

[23] Robert Serpollet et Pierre Yves Cochet « Hadamard Transforms for Parallel Digital Communications Applications to OFDM - DMT Modulations on Selective Channels » Traitement du Signal, Volume 14, N°3, pp : 276-283, 1997.

[24] Emmanuel Christophe « Compression des Images Hyperspectrales et son Impact sur la Qualité des Données » thèse de doctorat, L'école nationale supérieure de l'aéronautique et de l'espace, France, 2006.

[25] Donald Adjeroh, Tim Bell, and Amar Mukherjee, « The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching » Springer Science and Business Media, LLC, 2008.

[26] M. Burrows and D. J. Wheeler, « A block-sorting lossless data compression algorithm », Technical Report 124, Systems Research Center, Palo Alto, California, 10 may 1994.

[27] Juha Karkkainen, « Fast BWT in small space by blockwise suffix sorting » science direct, Theoretical Computer Science, vol: 387, pp: 249-257, 2007

[28] M.Salson, T.Lecroq, M.Léonard, L.Mouchard « A four-stage algorithm for updating a Burrows Wheeler transform » Theoretical Computer Science , Elsevier 2009.

[29] Elfitrin Syahrul, Julien Dubois, Vincent Vajnovszki, Taoufik Saidani, Mohamed Atri, « lossless image compression using burrows wheeler transform (methods and techniques) » IEEE International Conference on Signal Image Technology and Internet Based Systems (SITIS.2008.40), 2008

[30] R.Benzid, F.Marir, and N.-E. Bouguechal « Electrocardiogram Compression Method Based on the Adaptive Wavelet Coefficients Quantization Combined to a Modified Two-Role Encoder » IEEE signal processing letters, vol. 14, no. 6, June 2007.

[31] S. G. Hoggar « mathematics of digital images (Creation, Compression, Restoration, Recognition) » Cambridge University Press, 2006.

[32] www.petitcolas.net/fabien/image_database.

[33] ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus

Résumé :

La compression est l'un des volets de la théorie de l'information les plus répandus. Elle joue un rôle important dans l'optimisation du coût de transmission et/ou de stockage. De ce fait, nous avons présenté les différentes stratégies de compression sans et avec perte.

Dans ce travail, la transformée Burrows Wheeler (TBW) a été incluse comme phase de prétraitement dans la chaîne de compression des deux types de données (texte et image en niveaux de gris).

Les résultats obtenus sont considérés satisfaisants et l'utilisation d'autres transformées associées peut les améliorer encore plus.

Mots clé : Compression de données, Transformée Burrows Wheeler, Compression de texte, Compression des images niveaux de gris.

Abstract:

Data Compression constitutes one of the main categories belonging to the information theory. It plays an important role in optimization of the transmission and/or storage costs. In this context, we have presented several strategies, usually, used for lossless and lossy data compression.

The present work includes the Burrows-Wheeler transform (BWT), as a preprocessing step, in the data compression scheme for both text and gray level images.

Obtained results are considered satisfactory and the use of other associated transforms will improve them better.

Key words: Data Compression, (Burrows Wheeler Transform), Text compression, Gray level image compression.