Polycopié du cours : Codage et Théorie de l'information,

3ième Licence, Télécom, ST

Chapitre II : Codage de source

Contenu des Cours 1,2 , 3 et 4:

Préparé par : Mr Abdenour Hacine Gharbi, enseignant à l'université de Bordj Bou Arréridj

CONTENU DU COURS 1, 2, 3 ET 4:

Généralités sur le codage de source, codage d'une source discrète sans mémoire, classifications des codes de source, efficacité d'un code de source, théorème du codage de source, codage de Shannon-Fano, Codage de Huffman, codage par dictionnaire statique, codage par dictionnaire adaptatif, Algorithmes de Lempel-Ziv, Algorithme LZW. Algorithme du codage arithmétique.

Chapitre II : Codage de source

1. GENERALITES SUR LE CODAGE DE SOURCE

1.1. INTRODUCTION

En communication numérique, un canal de transmission se caractérise généralement par sa capacité de transmission qui est définie comme le débit d'information maximale supporté par le canal. Cette caractéristique du canal constitue une limitation de la quantité d'information que la source peut transmettre, ce qui rend difficile la transmission des messages générés par une source de débit d'information élevé. Par exemple, le débit binaire d'une source de séquence vidéo peut atteindre plus de 160 Mégabit par seconde (Mbps), ce qui rend difficile la transmission des séquences vidéo. Dans cet exemple, le débit peut être réduit en minimisant la redondance spatiale des pixels adjacents ainsi que la redondance temporelle des images successives. Cette opération de réduction appelée compression permet de réduire le débit binaire et de garantir une transmission rapide de la séquence vidéo.

Ainsi, il est nécessaire de représenter la sortie de la source d'information de toute nature (parole, vidéo, image, texte,…etc) sous une forme compacte en réduisant le nombre de bits requis pour la représentation du message à transmettre. Cette réduction ou compression peut être assurée par un codage permettant de convertir la sortie d'une source discrète en une séquence de symboles binaires (mot binaire) avec un débit binaire réduit. Ce codage appelé codage de source a pour objectif de réduire la redondance d'information de la source pour transmettre un débit de symbole minimum, le plus proche possible du débit d'information réel de la source.

Ce codage consiste à minimiser la longueur moyenne (nombre de bits moyen par symbole) des mots binaire associés aux symboles de l'alphabet de la source, en réduisant la redondance d'information émise par cette source (voir figure II.1). Il est également utilisé pour la compression des données ou signaux (vidéo, image, audio, texte,…etc) dans le but de réduire l'encombrement mémoire, et il est utilisé pour le cryptage d'informations pour les sécuriser.

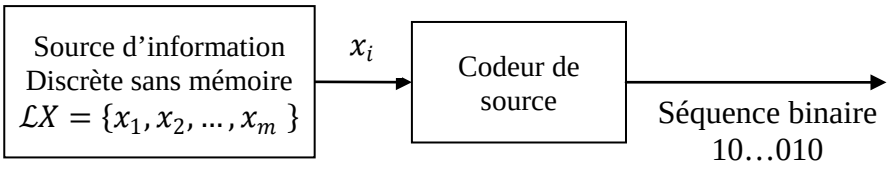

Figure. II.1 : Codage de source d'information

Le codage de source se base principalement sur le théorème de Shannon qui limite la longueur moyenne minimale des mots codés par la valeur de l'entropie de la source. On distingue principalement le codage entropique qui utilise des statistiques sur l'alphabet de la source d'information pour construire un code (l'ensemble des mots binaires). Ce codage consiste à concevoir un code à longueur variable de telle façon que sa longueur moyenne s'approche de l'entropie de la source.

Les algorithmes couramment utilisés pour le codage entropique sont l'algorithme de Shannon-Fano et l'algorithme de Huffman qui permettent de coder un message, symbole par symbole. Cependant, un autre algorithme de codage entropique appelé l'algorithme arithmétique n'effectue pas le codage symbole par symbole mais par une chaine de symboles. Un autre algorithme appelé algorithme de Lempel-ziv permet de remplacer une chaine de symboles d'un message par un code plus court qui est l'indice de cette chaine dans un dictionnaire. Cet algorithme appelé également algorithme à dictionnaire. Ces algorithmes seront détaillés dans les sections suivantes.

Dans ce chapitre, on s'intéresse au codage de source appliqué sur une source discrète sans mémoire.

1.2. CODAGE D'UNE SOURCE DISCRETE SANS MEMOIRE

Considérons une source d'information discrète sans mémoire (SDSM) X qui possède un alphabet $\mathcal{L}X = \{x_1, x_2, \dots, x_m\}$ dont la probabilité d'occurrence de chaque symbole x_i est $P(x_i)$ (avec $i = 1,2,\dots,m$.

Rappelons que l'entropie $H(X)$ de cette source est donnée par l'équation (I.59) (voir chapitre I):

$$H(X) = -\sum_{i=1}^{m} P(x_i) \log_2 P(x_i) \qquad \text{bit/symbole}$$

Supposons qu'un codeur code chaque symbole x_i par un mot binaire c_i du code. L'ensemble des mots c_i constitue le code $C = \{c_1, c_2, \dots, c_m\}$.

Soit n_i la longueur en bits du mot c_i. La longueur moyenne L des mots du code C est donnée par :

$$L = \sum_{i=1}^{m} P(x_i)\ n_i \quad \text{bit/symbole} \qquad (II.1)$$

La longueur moyenne L appelée également taux de codage représente le nombre moyen de bits par symbole de la source d'information.

1.2.1. Classification des codes

Les codes peuvent être regroupés en plusieurs classes selon la longueur des mots codés, la structure du code et la manière du décodage.

- **Code à longueur fixe** :

  Un code est dit à longueur fixe si tous les mots du code ont la même longueur.

- **Code à longueur variable :**

  Un code est dit à longueur variable si les mots du code n'ont pas une longueur fixée.

- **Code distinct (ou régulier)** :

  Un code distinct est un code pour lequel chaque mot est différent des autres mots du code.

- **Code à préfixe différent** :

  Un code est dit à préfixe différent si aucun mot ne peut être le préfixe d'un autre mot du code

- **Code à décodage unique (déchiffrable)** :

  Un code est dit à décodage unique si le décodeur reconstruit parfaitement et sans ambigüité la séquence originale générée par la source à partir de la séquence codée. Le décodeur doit interpréter la séquence codée d'une façon unique comme une succession de mots bien définis. Les codes à préfixe différent sont des codes à décodage unique. Cependant les codes à décodage unique ne sont pas toujours des codes à préfixe différents.

- **Code instantané** :

  Un code à décodage unique est dit code instantané si le décodeur peut reconnaitre la fin de tout mot du code sans examiner les symboles du mot suivant. Les codes instantanés se caractérisent par la propriété de préfixe différent. Ainsi, les codes instantanés sont des codes à préfixe différents.

- **Code optimal** :

Un code est dit code optimal s'il est un code instantané ayant une longueur moyenne minimale.

Une condition nécessaire et suffisante assurant l'existence d'un code à décodage unique instantané C de m mots codés en binaire, est donnée comme suit :

$$\sum_{i=1}^{m} 2^{-n_i} \leq 1 \qquad (II.2)$$

Où n_i est la longueur d'un mot codé c_i associé au symbole x_i générée par une source SDSM X.

Cette condition appelée l'inégalité de Kraft ne permet pas de vérifier si un code C est instantané.

1.2.2. Efficacité d'un code de source

Un code C est d'autant plus efficace que L est faible. Ainsi l'efficacité d'un code est relativement liée à la valeur minimum de L_{min} . L'efficacité notée par η est définie comme suite :

$$\eta = \frac{L_{min}}{L} \qquad (II.3)$$

Un code C est dit efficace si η approche 1. La redondance du code est donnée par :

$$\gamma = 1 - \eta \qquad (II.4)$$

1.2.3. Théorème du codage d'une source d'information (théorème de Shannon)

Le théorème du codage d'une source d'information montre que, pour une source discrète sans mémoire X ayant une entropie $H(X)$, la longueur moyenne L du nombre de bits par symbole est limitée par une borne inférieure :

$$L \geq H(X) \qquad (II.5)$$

Ainsi, L_{min} est égale à $H(X)$, l'efficacité peut s'écrire sous la forme suivante :

$$\eta = \frac{H(X)}{L} \qquad (II.6)$$

Donc, un code C est efficace si la longueur L peut approcher $H(X)$.

Exemple (II.1) :

Soit X une source discrète sans mémoire qui génère cinq symboles : x_1, x_2, x_3, x_4 et x_5 dont les probabilités sont données dans le tableau (II.1). Ce tableau illustre deux codes de source A et B. Le code A est un code à longueur fixe, alors que le code B est un code à longueur variable.

1. Calculer l'entropie $H(X)$ de la source
2. Calculer la longueur moyenne de chaque code.
3. Calculer l'efficacité et la redondance de chaque code
4. Quelle est le meilleur code ?
5. Est-ce que le code B est un code à décodage unique ? justifier

symbole x_i	$p(x_i)$	*Code A*	*Code B*
x_1	0.48	000	1
x_2	0.27	001	00
x_3	0.13	010	010
x_4	0.08	011	0110
x_5	0.04	100	0111

Tab. II.1 : Exemples de codes de source d'information

Solution :

1. L'entropie $H(X)$ est calculée comme suit:

$$H(X) = -\sum_{i=1}^{5} p(x_i).\log_2\big(p(x_i)\big)$$
$$= 1.8782\ bit/symbole$$

2. La longueur moyenne d'un code est donnée par l'équation (II.1):

$$L = \sum_{i=1}^{m} P(x_i)\ n_i$$

Pour le code A, la longueur de chaque symbole est fixée à 3, ainsi la longueur moyenne de ce code est égale à:

$$L_A = 3 \text{ bit/symbole}$$

Pour le cas du code B, les longueurs des symboles $x_1, x_2, x_3, x_4\ x_5$ sont respectivement égales aux : 1, 2, 3, 4, 4. En appliquant la relation (II.1), on obtient la longueur moyenne du code B :

$$L_B = 1.89 \text{ bit/symbole}$$

3. L'efficacité et la redondance du code *A* sont données respectivement par les équations (II.6) et (II.4):

$$\eta_A = \frac{H(X)}{L_A}$$
$$= 0.6261$$
$$= 62.61\ \%$$
$$\gamma_A = 1 - \eta_A$$
$$= 0.3739$$
$$= 37.39\ \%$$

L'efficacité et la redondance du code *B* sont données par :

$$\eta_B = \frac{H(X)}{L_B}$$
$$= 0.9938$$
$$= 99.38\%$$
$$\gamma_B = 1 - \eta_B$$
$$= 0.0062$$
$$= 0.62\ \%$$

4. Le code *B* est le meilleur code puisqu'il présente l'efficacité la plus élevée et la redondance la plus faible. On peut remarquer que les codes à longueur variable peuvent donner de meilleures performances que celles des codes à longueur fixe.
5. Le code *B* est un code à préfixe différent, donc c'est un code à décodage unique.

2. ALGORITHMES DE CODAGE DE SOURCE

Plusieurs algorithmes de codage de source ont été développés pour concevoir des codes de source. Les principaux algorithmes de ce codage sont : l'algorithme de Shannon-Fano, l'algorithme de Huffman, l'algorithme arithmétique et l'algorithme de Lempel-ziv. Les trois premiers algorithmes sont de type de codage entropique qui utilise les propriétés statistiques de la source pour approcher la longueur moyenne des symboles de l'entropie de la source. Le dernier algorithme est de type de codage à dictionnaire.

2.1. ALGORITHME DE SHANNON-FANO

Soit X une source SDSM possédant un alphabet $\mathcal{L}X = \{x_1, x_2, \dots, x_m\}$ dont la probabilité d'occurrence de chaque symbole x_i est $P(x_i)$ (avec $i = 1,2, \dots, m$).

La procédure de construction d'un code de source par l'algorithme de Shannon-Fano est décrite comme suit :

1. Trier la liste des symboles par ordre décroissant des probabilités d'occurrence.
2. Diviser la liste triée en deux sous groupes ayant des sommes de probabilités les plus proche possibles.

3. Affecter un '0' pour le premier sous groupe (en haut) et un '1' pour le deuxième sous groupe (en bas).
4. Répéter les étapes 2 et 3 jusqu'à ce que chaque sous groupe contient qu'un seul élément (symbole).

Exemple (II.2) :

Considérons une source SDSM X de cinq symboles: x_1, x_2 , x_3, x_4 et x_5, dont les probabilités associées sont données dans le tableau suivant :

1. Construire un code de Shannon–Fano pour X.
2. Calculer l'entropie $H(X)$
3. Calculer l'efficacité η et la redondance γ de ce code.

Symbole	$p(x_i)$
x_1	0.25
x_2	0.45
x_3	0.15
x_4	0.10
x_5	0.05

Tab. II.2 : Probabilités d'occurrences des symboles de la source X

Solution

1. La construction du code de Shannon-Fano de la source X est illustrée dans le tableau suivant :

x_i	$p(x_i)$	*Etape 1*	*Etape 2*	*Etape 3*	*Etape 4*	*code*
x_2	*0.45*	*0*				*0*
x_1	*0.25*	*1*	*0*			*10*
x_3	*0.15*	*1*	*1*	*0*		*110*
x_4	*0.10*	*1*	*1*	*1*	*0*	*1110*
x_5	*0.05*	*1*	*1*	*1*	*1*	*1111*

Tab. II.3 : Construction du code de Shannon-Fano de la source X

2. L'entropie $H(X)$ est donnée par :

$$H(X) = -\sum_{i=1}^{5} p(x_i).\log_2(p(x_i))$$
$$= 1.9772\ bit/\ symbole$$

3. L'efficacité est donnée par :

$$\eta = \frac{H(X)}{L}$$

Où Lest la longueur moyenne du code.

L est donnée par :

$$L = \sum_{i=1}^{5} p(x_i).n_i \quad \text{, où } n_i \text{ est longueur du symbole } x_i$$
$$= 2\ bit/symbole$$
$$\eta = \frac{1.9772}{2}$$
$$= 0.9886$$
$$= 98.86\ \%$$

La redondance γ est donnée par :

$$\gamma = 1 - \eta$$
$$= 0.0114$$
$$= 1.14\%$$

On peut remarquer que le codage de Shannon-Fano permet de coder les symboles plus probables par des mots de code courts et les symboles peu probables par des mots de code longs. De plus, l'ensemble des mots constituent un code à préfixe différent.

Le code de Shannon-Fano est un code sous-optimal en termes de longueur des mots de code.

2.2. ALGORITHME DE HUFFMAN

L'algorithme de codage de Huffman a été développé à partir du codage de Shannon-Fano dans le but d'assurer l'optimalité du code.

Considérons une source SDSM X produisant m symboles $x_1, x_2, \dots, x_m$, dont les probabilités d'occurrence associées sont $P(x_i)$ (avec $i = 1,2, \dots, m$).

Les différentes étapes de l'algorithme du codage de Huffman sont comme suit :

1. Trier la liste des symboles par ordre décroissant des probabilités d'occurrence.
2. Regrouper les deux symboles de probabilités les plus faibles, ensuite calculer la somme de leurs probabilités et réordonner la liste réduite de la même façon de l'étape 1 en tenant compte ce groupe. Répéter cette étape de réduction de niveau 1 jusqu'au niveau où il ne reste qu'un couple de probabilités triées en ordre décroissant.
3. En commençant la procédure de codage au dernier niveau, affecter un '0' à la première probabilité (grande) et un '1' à la deuxième probabilité (faible).
4. Descendre d'un niveau de réduction (avec retour arrière) et affecter un '0' ou un '1' aux probabilités de la même manière de l'étape 3, ensuite concaténer cette affectation avec les affectations obtenues à l'étape 3.
5. Refaire l'étape 4 jusqu'au premier niveau de réduction.

Exemple (II.3) :

Reprenons l'exemple (II.2).

1. Construire un code de Huffman pour la source X.
2. Calculer l'efficacité η et la redondance γ de ce code.

Solution :

1. La figure (II.3) illustre les étapes de construction d'un code de Huffman pour la source X.
2. L'entropie de la source X est égale à: $H(X) = 1.9772\ bit/\ symbole$ (voir l'exemple (II.2).

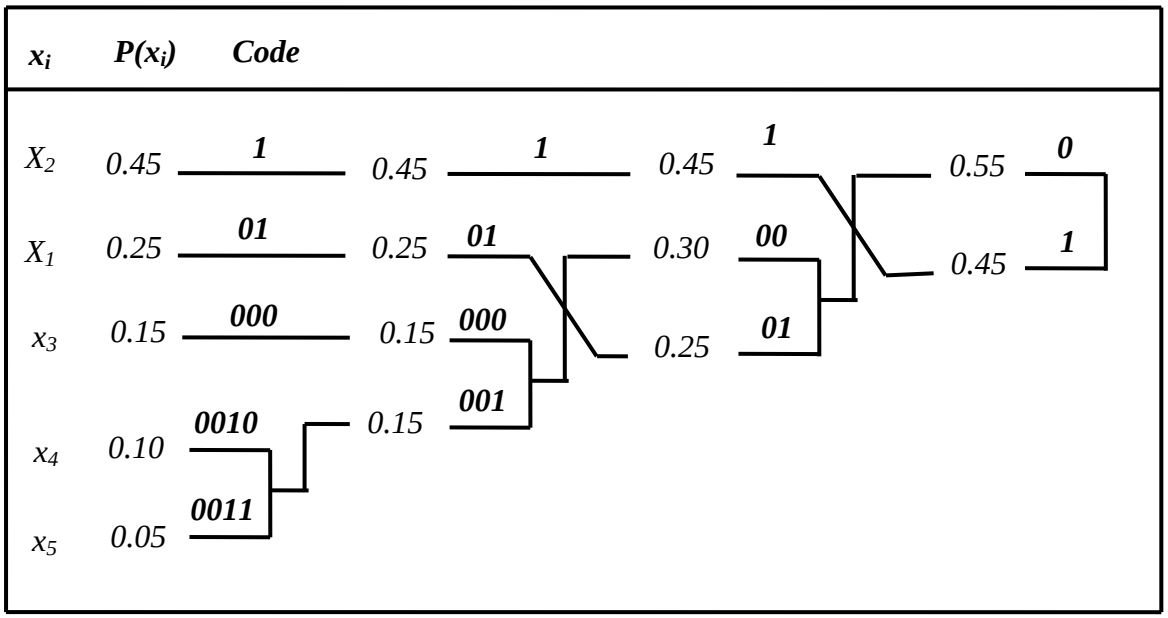

Figure. II.2 : Construction du code de Huffman de la source *X*

Les longueurs n_i (i=1,…,5) associées aux symboles x_i sont égales respectivement à 2, 1, 3, 4 et 4.

Ainsi, la longueur moyenne est donnée par :

$$L = \sum_{i=1}^{5} p(x_i).n_i$$
$$= 2\ bit/symbole$$

L'efficacité est calculée comme suit :

$$\eta = \frac{1.9772}{2}$$
$$= 0.9886 = 98.86\ \%$$

La redondance γ est donnée ainsi par :

$$\gamma = 1 - \eta$$
$$= 0.0114 = 1.14\%$$

2.3. ALGORITHME DU LEMPEL-ZIP

2.3.1. Introduction au codage par dictionnaire

Les algorithmes décrits précédemment supposent que la source génère des symboles indépendants. Ils permettent de coder une séquence de symboles (lettres) ou un message, symbole par symbole en formant la séquence codée par la concaténation des mots du code correspondant aux symboles du message.

Dans cette section on décrira un type d'algorithmes de codage de source qui tient en compte la structure de la séquence des symboles à coder dans le but de réduire le débit binaire. Ce type d'algorithmes consiste à construire un dictionnaire formé d'une liste de chaines ou groupes de symboles (appelés facteurs) et de coder chaque groupe (facteur) par son indice dans le dictionnaire. Ces algorithmes appelés 'algorithmes par dictionnaire' permettent ainsi de coder une séquence de symboles en remplaçant les facteurs constituant

cette séquence par leurs indices. Le codage de la séquence consiste ainsi à concaténer les mots de code des indices de facteurs au lieu des mots de code des symboles. Ces algorithmes sont plus efficaces avec des sources qui génèrent un petit nombre de facteurs assez fréquents, telles que des sources de texte et d'images. Le codage de texte consiste à construire un dictionnaire composé d'une liste de chaines de caractères (liste de mots ou facteurs) et de coder une séquence de caractères d'entrée par une séquence d'indices des facteurs constituant cette séquence. La séquence codée est constituée ainsi d'une séquence des mots de code des indices. Ce codage peut réduire considérablement le nombre de bits correspondant à la séquence de caractères, si la source génère des chaines de caractères apparues fréquemment dans la séquence. Les algorithmes de codage par dictionnaire peuvent être de type statique ou dynamique (adaptatif).

2.3.2. Codage par dictionnaire statique

Le codage par dictionnaire statique est généralement approprié pour certaines sources particulières dont les mots ou les chaines de caractères les plus probables sont connues préalablement. L'exemple d'un codage par dictionnaire statique bi-gramme utilise un dictionnaire dont chaque mot est constitué soit d'un caractère ou un pair de caractères. Le codeur lit un pair de caractère et vérifie ensuite si ce pair existe dans le dictionnaire. Si ce pair existe, alors son indice est codé et l'opération se répète en lisant le pair suivant. Si le pair de caractère n'existe pas, alors seulement l'indice du premier caractère est codé. Le deuxième caractère devient le premier caractère du pair suivant. Le codeur lit ensuite le caractère suivant pour compléter le pair et répète la procédure de vérification et de codage jusqu'à le dernier pair de la séquence de caractères.

Exemple (II.4) :

Considérons un source d'un alphabet de 5 caractères $\{a, b, c, d, r\}$.

Soit la séquence de caractères T donnée par : T='abracadabra'.

Un dictionnaire statique est construit à partir des connaissances à priori sur la source. Le tableau (II.4) illustre ce dictionnaire ainsi que les mots de code correspondant aux indices des caractères et des bi-grammes (facteurs) constituant ce dictionnaire.

Entrée (facteur)	Indice	Mot du code
A	1	000
B	2	001
C	3	010
D	4	011
R	5	100
Ab	6	101
Ac	7	110
Ad	8	111

Tab (II. 4) : dictionnaire statique (bi-grammes)

Le déroulement de la procédure du codage par dictionnaire statique de la séquence T est illustré sur la figure (II. 3).

Donc, la séquence de caractères T est codée par la séquence de mots de code Tc suivante :

Tc='**101100110111101100000**'. La séquence des indices est Ti=' 6 5 7 8 6 5 1 '.

Le codage binaire sur 3 bits pour chaque caractère de l'alphabet permet de coder la séquence d'entrée sur 33 bits (11 caractères), alors que le codage par dictionnaire statique permet de coder la séquence sur 21 bits. Ainsi, ce dernier codage conduit à un gain de compression G_c donnée par:

$$G_c = \frac{33 - 21}{33}$$
$$= 0.3636$$
$$= 36.36\%$$

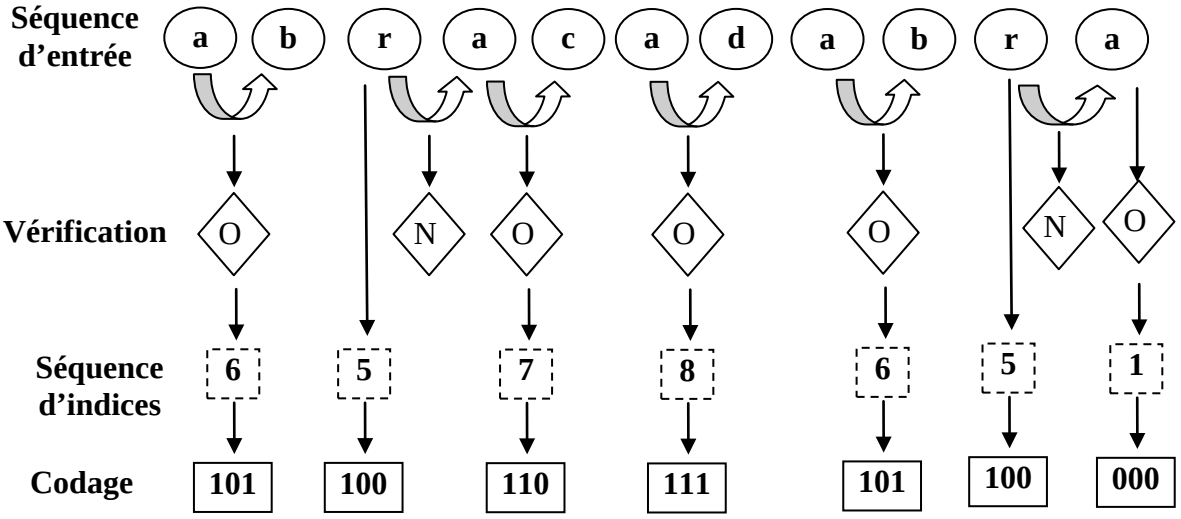

Figure (II.3) : Déroulement du processus de codage de la séquence de caractères par l'algorithme à dictionnaire statique

2.3.3. Codage par dictionnaire adaptatif

L'algorithme de codage par dictionnaire adaptatif consiste à construire dynamiquement la liste des mots du dictionnaire au cours du processus de codage. La plupart des algorithmes de ce type codage se basent sur deux papiers scientifiques de Jacob Ziv et Abraham Lempel, publiés en 1977 et 1978. Ces deux papiers proposent deux approches différentes pour la construction adaptative de dictionnaires. Ces deux approches connues sous le nom Lempel-Ziv ont conduit à un nombre de variations et à deux familles d'algorithmes appelées respectivement famille LZ77 (ou LZ1) et famille LZ78 (ou LZ2). La modification de l'algorithme de codage LZ78, la plus connue est appelée LZW (Lempel-Ziv-welsh).

2.3.4. Algorithme de codage LZW

L'algorithme de codage adaptatif LZW développé par Terry Welch en 1984, constitue une amélioration de l'algorithme LZ78. Cet algorithme est couramment utilisé dans les formats d'image Tiff, Gif.

Le principe de l'algorithme de codage LZW est comme suit :

1. On lit un caractère d'entrée *c* et on l'ajoute à la chaine (ou mot) courante M pour obtenir un nouvelle chaine (M//c) (au début : la chaine M est vide).
2. Si la nouvelle chaine (M//c) est présente dans le dictionnaire, on recommence l'opération en lisant le caractère suivant.
3. Si la nouvelle chaine (M//c) n'est pas présente dans le dictionnaire, on l'ajoute au dictionnaire et on code la chaine M (sans le caractère d'entrée c) par son indice. Ensuite la chaine courante M est remplacée par le dernier caractère lu c.
4. On recommence les étapes précédentes (1,2 et3) jusqu'à la fin de la séquence d'entrée.

Les détails de l'algorithme de codage LZW sont donnés comme suit :

Algorithme LZW :

Soit chaine ← ∅ la chaine courante (initialisation)

Tant que on n'est pas à la fin de la séquence d'entrée **faire**

Lire un caractère c de la séquence

Si chaine//c est dans le dictionnaire **Alors**

chaine ← chaine//c

Sinon

Afficher le mot du code de la chaine

Ajouter chaine//c au dictionnaire

Chaine ← c

Fin Si

Fin Tant que

Exemple (II. 5) :

Soit le texte suivant : T= '**aabaacaacba**'

- Construire le dictionnaire et générer la séquence d'indices qui correspond à ce texte en utilisant l'algorithme LZW.

Solution :

Supposons que le dictionnaire initial est illustré dans le tableau suivant :

Entrée (facteur)	Indice
a	1
b	2
c	3

Tab (II.5) : dictionnaire initial du codage LZW

Le déroulement du codage du texte T en utilisant l'algorithme LZW est illustré sur le tableau (II.6). Selon le résultat de codage, la séquence des indices est : T_i= ' 1 1 2 4 3 7 6 '.

Etape	Chaine courante	Caractère lu	Affichage de l'indice	Ajout au dictionnaire	
				Chaine ajoutée	Indice de la chaine ajoutée
0		a			
1	a	a	1	aa	4
2	a	b	1	ab	5
3	b	a	2	ba	6
4	a	a			
5	aa	c	4	aac	7
6	c	a	3	ca	8
7	a	a			
8	aa	c			
9	aac	b	7	aacb	9
10	b	a	6		

Tab (II.6) : déroulement du processus de codage du texte T
par l'algorithme LZW

2.4. ALGORITHME DU CODAGE ARITHMETIQUE

L'algorithme arithmétique est un algorithme de codage statistique généralement plus optimal que le codage de Huffman. Cet algorithme consiste à coder une chaine de symboles par un seul nombre en virgule flottante, compris entre 0 et 1. En codage de Huffman, chaque symbole est codé par un nombre entier de bits, alors qu'en codage arithmétique un symbole est codé en moyenne par un nombre réel de bits (éventuellement inférieur à un bit).

Exemple (II.6) :

Soit X une source SDSM produisant trois symboles x_1, x_2, x_3, dont les probabilités d'occurrence associées sont respectivement égales à : $P(x_1) = 0.94$, $P(x_1) = 0.04$ et $P(x_1) = 0.02$.

L'entropie de cette source égale à $H(X) = 0.3825$ bit \symbole. Le codage par l'algorithme de Huffman

permet d'associer respectivement aux symboles x_1, x_2, x_3 les mots de code : 0, 10 et 11. Ce codage permet d'obtenir ainsi un code de longueur moyenne égale à 1.06 bit \symbole avec une efficacité de 36,09% et une redondance de 63,91%. Cette faible efficacité peut être justifiée par la méthode du codage de Huffman qui exige de coder chaque caractère par un nombre entier de bits (au minimum un bit). Ainsi, cet exemple nous montre qu'il faut un autre algorithme permettant de coder des symboles sur un nombre réel de bits (possiblement inférieur à un bit). Ce problème peut être surmonté en utilisant l'algorithme du codage arithmétique.

2.4.1. Principe du codage arithmétique

Soit X une source discrète d'un alphabet de *N* symboles $\{x_1, x_2, \dots, x_N\}$, dont les probabilités d'occurrence associées sont respectivement égales à : $P(x_1) = p_1$, $P(x_2) = p_2, \dots$, $P(x_N) = p_N$.

Considérons une séquence de *M* symboles *S*='$s_1, s_2, \dots, s_M$' constituée des symboles de l'alphabet de X.

Le codage arithmétique permet, à partir des probabilités d'occurrence des symboles de la source, de coder la séquence de symboles *M* par un mot de code formé d'un nombre réel représenté en virgule flottante, et appartenant à l'intervalle [0, 1[.

Le mot du code de la séquence de symboles est construit récursivement par subdivision d'intervalles. Initialement, l'intervalle [0,1[est subdivisé en *N* sous intervalles dont chacun est associé à un symbole de l'alphabet. Les bornes de chaque sous intervalles sont déterminés à partir des probabilités des symboles. A chaque itération *j*, l'intervalle correspondant au symbole s_{j-1} lu à l'itération *j*-1 est subdivisé en *N* sous intervalles en fonction des probabilités des symboles. Le sous intervalle correspondant au symbole s_j lu à l'itération *j* est sélectionné pour effectuer la même opération de subdivision à l'itération *j*+1. Tout nombre réel appartenant au sous intervalle de la dernière itération ($N^{ième}$ itération) peut représenter la séquence à coder.

Procédure du codage arithmétique:

L'intervalle [0, 1[est considéré comme l'intervalle de variation de la fonction cumulative $F(x_i)$ des probabilité des symbole $\{x_i \backslash i = 1:N\}$ déterminée récursivement comme suit :

$$F(x_i) = F(x_{i-1}) + p_i \ , \ \ i = 1:N \ \text{ avec } \ F(x_i) = 0 \text{ pour } i < 1 \qquad (II.7)$$

On associe à chaque symbole x_i l'intervalle initial $[F(x_{i-1}), F(x_i)[$. On note les bornes initiales inférieure et supérieure $F(x_{i-1})$ et $F(x_i)$ respectivement par $f_i \ et \ h_i$.

$$f_i = F(x_{i-1}) \ , \qquad h_i = F(x_i) \ , \ \ i = 1:N \qquad (II.8)$$

Les étapes du codage arithmétiques sont comme suit :

1. L'étape initiale :

   Initialiser le premier intervalle avec deux bornes : Borne inférieure $LB = 0$ et borne supérieure $HB = 1$. La taille de cet intervalle est donnée par : $L = HB - LB = 1$.

2. Subdiviser cet intervalle en *N* sous intervalles T_i de la forme suivante $[Lsb_i, Hsb_i[$. Cette subdivision

est effectuée comme suit :

$$Lsb_i = LB + L.f_i,\ \ Hsb_i = LB + L.h_i\ \ , i = 1:N \qquad (II.9)$$

- Associer à chaque sous intervalle T_i le symbole x_i.

3. Lire le premier symbole s_1 de la chaine S et sélectionner l'intervalle $T = [Lsb, Hsb[$ qui correspond à ce symbole.
4. Redéfinir l'intervalle précédent $[LB, HB]$ ainsi que sa taille L comme suit :

$$LB = Lsb\ ,\quad HB = Hsb$$

$$L \leftarrow HB - LB$$

5. Refaire les étapes 2,3 et 4 jusqu'à la fin de la séquence.
6. Retourner le dernier intervalle $[LB, HB]$.

Tout nombre appartenant au dernier intervalle peut représenter la séquence à coder.

Cette procédure peut être simplifiée en effectuant à chaque itération *j*, la lecture d'un symbole s et le recalcule des bornes du sous intervalle de ce symbole. Cet sous intervalle devient ensuite l'intervalle à subdiviser à l'itération *j*+1.

Les détails de l'algorithme du codage arithmétique sont donnés comme suit :

Algorithme du codage arithmétique :

Soit $HB \leftarrow 0$

Soit $LB \leftarrow 0$

Tant que on n'est pas à la fin de la séquence **Faire**

Lire un symbole s de la séquence.

Soient f, h les bornes inférieure et supérieure de l'intervalle initial correspondant à s (données par la relation (II.8).

$L \leftarrow HB - LB$ (redéfinition de la taille de l'intervalle précédent)

$HB \leftarrow LB + L * h$ (recalcule de la borne supérieure de l'intervalle prochain)

$LB \leftarrow LB + L * f$ (recalcule de la borne inférieure de l'intervalle prochain)

Fin Tant que

Retourner les bornes LB et HB.

Exemple (II. 6) :

Soit une source discrète X d'un alphabet de trois symboles a, b, c dont les probabilités d'occurrence associées sont respectivement égales à : $P(a) = 0.2$, $P(b) = 0.4$ et $P(c) = 0.4$.

Soit la séquence *M=*'*abcaa*'.

- Coder la séquence *M* par l'algorithme de codage arithmétique.

Solution:

Les intervalles initiaux associés aux trois symboles sont illustrés sur le tableau tab (II.7).

Symboles	a	b	c
probabilité	0.2	0.4	0.4
Intervalle initial	[0, 0.6[	[0.6, 0.8[	[0.8,1[

Tab II.7 : Intervalles initiaux associés aux symboles

Le déroulement du processus du codage arithmétique est illustré sur la le tableau (II.8).

Symbole lu	Taille L	Borne LB	Borne HB
a	1	0	0.6
b	0.6	0.36	0.48
c	0.12	0.4560	0.48
a	0.024	0.4560	0.4704
a	0.0144	0.4560	0.46464

Tab II.8 : Déroulement du processus du codage arithmétique de la chaine M='abcaa'

Ce déroulement peut être illustré graphiquement sur la figure suivante :

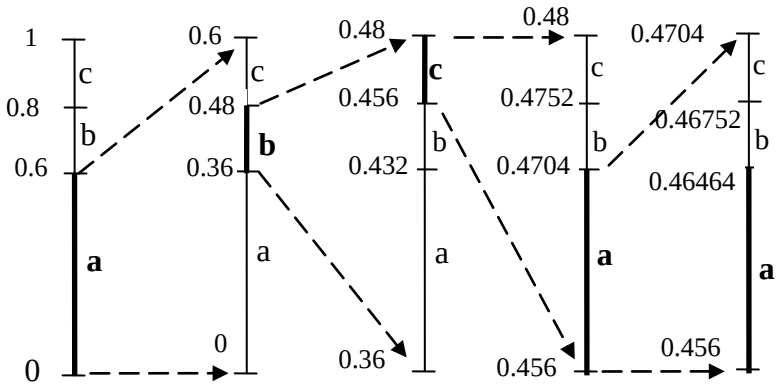

Figure. II.4 : Déroulement du processus du codage arithmétique de la chaine M='abcaa'

Tout nombre appartenant à l'intervalle [0.456, 0.46464] peut représenter la séquence *M*. choisissons par exemple le nombre 0.46032 (au milieu de l'intervalle).