Cours/TD 5 Codage Shannon. Codage arithmétique : Elias

5.1 De l'algorithme de Fano et Shannon au codage arithméthique

Si

$$\sum_{j=0}^{N-1} 2^{-l_j} < 1 \text{ l'inégalité de Kraft-McMillan}$$

est vérifiée, alors il existe un code uniquement décodable et même préfixe, qui admette $l_1, l_2, ..., l_{N-1}$ comme distribution de longueurs. Un algorithme simple qui donne un code préfixe $c_1, ..., c_{N-1}$ à partir d'une distribution de longueurs $l_1, l_2, ..., l_{N-1}$ vérifiant l'inégalité de Kraft-McMillan est : *a chaque mot de code c_i (à trouver) on associe le nombre $= 0, \overline{c_i} \in [0, 1[$ dont les décimales de l'écriture en base 2 est formée des bits de c_i. On note I_i l'intervalle $I_i = [\overline{c_i}; \overline{c_i} + 2^{-l_i}[$.*

Exemple. *$c_i = 010$ donne $\overline{c_i} = 0, \overline{010} = \frac{1}{4}$. et $I_i = [0,010; 0,011[= [\frac{1}{4}; \frac{3}{8}[$ est l'ensemble des nombres de $[0; 1[$ dont les décimales en base 2 commencent par c_i.*

Le mot code c_i détermine uniquement l'intervalle I_i : tronquer le développement binaire de la limite gauche de l'intervalle (de longueur l_i) à $\log_2 \frac{1}{l_i}$ bits pour retrouver c_i. La condition de code préfixe deviennent : les I_i sont des intervalles disjoints. Donc l'inégalité de Kraft-McMillan dit que la somme des longueurs des I_i est strictement plus petite que 1. S'ils sont disjoints, on peut donc les mettre bout-à-bout en restant dans le segment $[0, 1[$. Ainsi un algorithme de construction du code instantané sera : On met bout-à-bout les intervalles I_i, classés par longueurs décroissantes (l_i croissantes) dans le segment $[0, 1[$ en partant de la gauche.

I1 I2 I3 ....

———I————I———I————— 0.0 0.10 0.110 0.111

Au debut $\overline{c_0} = 0.0...0$. En general à chaque étape on rajoute à c_{i-1} un 1 et on obtient $\overline{c_i}$ l' extrémité droite de I_i. On peut compléter avec des zéros si nécessaire pour obtenir une longueur $l_{i+1} > l_i$.

Exemple. *Soit une source discrète sans mémoire codée avec des mots code avec de longueurs* l_i

l_i	*code*
1	0
2	10
3	110
5	11100
5	11101
5	11110
6	111110
6	111111

Noter qu'avec cet algorithme on détecte automatiquement si si les li ne vérifient pas l'inégalité de Kraft-McMillan : on "dépasse" alors l'extrémité droite du segment $[0, 1[$*, et on ne peut plus continuer comme dans :*

l_i	*code*
1	0
2	10
3	110
4	1110
5	11110
5	11111
6	*Erreur*

$1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/32 + 1/64 > 1$

Soit une source discrete et sans mémoire produisant (indépendament) des symboles (lettres/mots) d'un alphabet $a_0, a_1, \cdot, a_{N-1}$ selon la distribution de probabilité $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$. Supposons que les symboles de l'alphabet sont déjà ordonnés par probabilité décroissante. $p_0 \geq p_1 \geq \cdots \geq p_{N-1}$.

Idée naturelle (Shannon et Fano) était d'associer aux symboles de notre alphabet des mots code binaires dont la longueur est égale au contenu d'information des symboles à coder.

Pour un symbol a_j avec une probabilités d'apparition p_j qui n'est pas une puissance binaire ce contenu d'information est $I(a_j) = -\log_2 p_j$ et Shannon demande que la longueur de bits necessairés soit $l_j = \lceil I(a_j) \rceil = \lceil -\log_2 p_j \rceil$.

L'interprétation géométrique de l'algorithme de Fano :

Par la suite d'une dichotomie binaire réittérée sur [0, 1 [l'algorithme de Fano trouvait une partition d'intervalles. La dichotomie d'un interval s'arrêtait quand l'intervalle $[A_j, A_{j+1}[$ avait la longueur égale à la probabilité 2^{-l_j} de (la production de) du symbol a_j (i.e. un seul

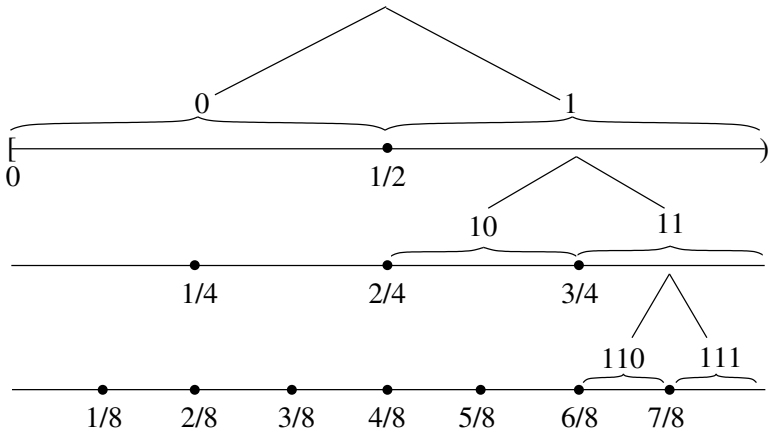

FIGURE 5.1 – Arbre partiel de dichotomie binaire réitéré de $[0, 1)$ pour $p = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right)$

element par intervalle). Le mot code était le développement binaire fini de la borne inférieure A_j de cet intervalle. La précision de ce mot était exactement $l_j = -\log_2 p_j$ – voir figure 5.1.

L'algorithme simplifié de Fano peut être lui aussi generalisé mais le code produit n'est pas unique, ni optimal.

1. ordonner les symboles de l'alphabet sur une colonne par probabilité décroissante.
2. diviser l'ensemble des symboles en deux sous-ensembles de probabilités cumulées presque égales.
3. coder avec "0" (resp. "1") les éléments du premier (resp. deuxième) sous-ensemble
4. continuer de manière récursive jusqu'au moment où tous les sous-ensembles ne comportent plus qu'un élément

D'une manière plus generale l'algorithme de Shannon associe aussi d'une manière biunivoque :

– aux symboles de l'alphabet une partition d'intervalles dans $[0, 1[$
– à un symbol un mot code qui est un nombre réel dans l'intervalle correspondant.

5.2 Algorithme de Shannon

L'interprétation geometrique de l'algorithme de Shannon

est de trouver une partition d'intervalles avec la propriété que la longueur de chaque intervalle $[A_j, A_{j+1}[$ est égale à la probabilité de (la production de) du symbol a_j : $p_j = A_{j+1} - A_j$, $0 \leq j \leq N - 1$.

$A_0 = 0$
$A_1 = p_0$
$A_2 = p_0 + p_1$

$A_3 = p_0 + p_1 + p_2$

$\vdots$

$A_N = p_0 + p_1 + \cdots + p_{N-1} = 1$

Un mot code c_j est associe à l'intervalle $[A_j, A_{j+1}[\subset [0,1[$ de la manière suivante :

$$c(A_j, A_{j+1}) = \alpha_1\alpha_2 \ldots \alpha_{l_j} \quad \Longleftrightarrow \quad A = 0.\alpha_1\alpha_2 \ldots \alpha_{l_j} * \quad (\text{ début du développement binaire du nombre réel } A_j \text{ }), \text{ avec } l_j = \lceil -\log_2(p_j) \rceil.$$

On prend autant de bits de ce développement que le "contenu d'information de l'intervalle" l_j le dicte.
Maintenant pour assurer la correspondance biunivoque il faut prouver que le mot code associe a un intervalle $[A_j, A_{j+1}[$ est unique. Plus on peut montrer que :

Lemme 1. *Le code Shannon est préfixe, i.e. le mot code $c_j = c(A_j, A_{j+1})$ ne peut pas être un préfixe du mot $c_{j+1} = c(A_{j+1}, A_{j+2})$, pour $0 \leq j \leq N-2$.*

Preuve. *Supposons que le codage binaire de A_j a la longueur $l_j = \lceil -\log_2 p_j \rceil$.*
Réécrivons les conditions de Shannon :

$$l_j - 1 < -\log_2 p_j \leq l_j, 0 \leq j \leq N-1$$
$$\Leftrightarrow 2^{-l_j} \leq p_j < 2 \cdot 2^{-l_j}, 0 \leq j \leq N-1$$

.

Par construction $A_k \geq A_{j+1} = A_j + p_j \geq A_j + \frac{1}{2^{l_j}}$. Donc $A_k - A_j \geq \frac{1}{2^{l_j}}$, i.e. leurs développements binaires sont différents sur les premiers l_j positions. Par consequence le mot code c_j n'est pas un préfixe du mot code c_k , $j < k$.

Remarque. *On note que on peut dire aussi que le codage Shannon-Fano associe au symbol a_j le bits du plus petit (i.e. celui avec moins de bits) nombre binaire rationnel de l'interval $[A_j, A_{j+1}[$.*

Algorithme de Shannon

1. ordonner les symboles de l'alphabet sur une colonne par probabilité décroissante.
2. calculer $A_0 = 0$ et $A_{j+1} = A_j + p_j$
3. pour chaque $0 \leq j \leq N-1$ écrire le développement dyadique $A = 2 \cdot A_j$ jusqu'au $l_j = \lceil -\log_2 p_j \rceil$ "décalage de la virgule "
   - si $A \geq 1$, alors $\alpha_1 = 1$ et $A = 2A - 1$
   - si $A \leq 1$, alors $\alpha_1 = 0$ et $A = 2A$
4. afficher $A_j = 0.\alpha_1\alpha_2\alpha_3\alpha_4 \ldots \alpha_{l_j}$

Exemple. *Trouver le développement binaire $a = 0.\alpha_1\alpha_2\alpha_3\alpha_4 \ldots$ de $a = \frac{1}{11}$.*

$2a = \frac{2}{11} < 1 \Longrightarrow \alpha_1 = 0$, $4a = \frac{4}{11} < 1 \Longrightarrow \alpha_2 = 0$

$8a = \frac{8}{11} < 1 \Longrightarrow \alpha_3 = 0$, $16a = \frac{16}{11} = 1 + \frac{5}{11} \Longrightarrow \alpha_4 = 1$

$a^{(4)} = \frac{5}{11} = 0.\alpha_5\alpha_6\alpha_7\ldots$, $2a^{(4)} = \frac{10}{11} < 1 \Longrightarrow \alpha_5 = 0$

$4a = \frac{20}{11} = 1 + \frac{9}{11} \quad \Longrightarrow \quad \alpha_6 = 1$

$a^{(6)} = \frac{9}{11} = 0.\alpha_7\alpha_8\alpha_9\ldots$, $2a^{(6)} = \frac{18}{11} = 1 + \frac{7}{11} \Longrightarrow \alpha_7 = 1$

$a^{(7)} = \frac{7}{11} = 0.\alpha_8\alpha_9\alpha_{10}\ldots$, $2a^{(7)} = \frac{14}{11} = 1 + \frac{3}{11} \Longrightarrow \alpha_8 = 1$

$a^{(8)} = \frac{3}{11} = 0.\alpha_9\alpha_{10}\alpha_{11}\ldots$, $2a^{(8)} = \frac{6}{11} < 1 \Longrightarrow \alpha_9 = 0$

$4a^{(8)} = \frac{12}{11} = 1 + \frac{1}{11} \quad \Longrightarrow \quad \alpha_{10} = 1$

$a^{(10)} = \frac{1}{11} = 0.\alpha_{11}\alpha_{12}\alpha_{13}\cdots = a$.

Le développement binaire (10-périodique) de $a = \frac{1}{11} = 0.\overline{0001011101}$

Exercice. *Trouver les mots code $c(A, B)$ suivants :*

1. $c\left(\frac{1}{11}, \frac{1}{5}\right)$
2. $c\left(\frac{3}{8}, \frac{1}{2}\right)$
3. $c\left(\frac{1}{12}, \frac{7}{8}\right)$
4. $c\left(\frac{3}{5}, \frac{3}{4}\right)$
5. $c\left(\frac{1}{7}, \frac{1}{6}\right)$

1. $c\left(\frac{1}{11}, \frac{1}{5}\right)$

Premier pas : Determiner la precision du développement dyadique. La longueur du codage est donnée par l'information qui correspond à l'interval $B-A = \frac{1}{5}-\frac{1}{11} = \frac{6}{55}$. Donc $\lceil \log_2 \frac{55}{6} \rceil = 4$, parce que $8 < \frac{55}{6} < 16$.

$$\frac{1}{11} = 0.\alpha_1\alpha_2\alpha_3\alpha_4\cdots$$

Deuxième pas : Obtenir le développement binaire du nombre $A = \frac{1}{11}$ qui est entre 0 et 1. Donc

$$\frac{1}{11} = 0.0001\cdots \Longrightarrow c\left(\frac{1}{11}, \frac{1}{5}\right) = \alpha_1\alpha_2\alpha_3\alpha_4\cdots = 0001$$

2. $c\left(\frac{3}{8}, \frac{1}{2}\right)$

Premier pas : Determiner la precision du développement dyadique. La longueur du codage est donnée par l'information qui correspond à l'interval $B - A = \frac{1}{8}$. Donc $\log_2 8 = 3$.

$$\frac{3}{8} = 0.\alpha_1\alpha_2\alpha_3\cdots$$

Deuxième pas : Obtenir le développement binaire du nombre $A = \frac{3}{8}$ qui est entre 0 et 1.

$$2 \times \frac{3}{8} = \alpha_1.\alpha_2\alpha_3\cdots$$

$$2 \times \frac{3}{8} = \frac{6}{8} < 1 \Longrightarrow \alpha_1 = 0 \text{ et } \frac{6}{8} = 0.\alpha_2\alpha_3\cdots$$

$$2 \times \frac{6}{8} = \alpha_2.\alpha_3\cdots$$

$$2 \times \frac{6}{8} = \frac{12}{8} = 1 + \frac{4}{8} = 1 + \frac{1}{2} \geq 1 \Longrightarrow \alpha_2 = 1 \text{ et } \frac{1}{2} = 0.\alpha_3\cdots$$

$$2 \times \frac{1}{2} = \alpha_3$$

$$2 \times \frac{1}{2} = 1 \Longrightarrow \alpha_3 = 1 \Longrightarrow c\left(\frac{3}{8}, \frac{1}{2}\right) = \alpha_1\alpha_2\alpha_3\cdots = 011$$

Exercice. *Déterminer tous les intervalles* $[A, B[$ *tels que* $c(A, B) = 10101$.

(Chercher d'abord l'intervalle le plus naturel déterminé par 10101*, puis réfléchir dans quelle mesure ceci est "déformable" sans effet sur le mot code).*

Exercice. *Considérons la source qui produit les huit lettres* $a_0, a_1, \ldots, a_7$ *selon la distribution de probabilité* $\mathbf{p} = (p_0, p_1, \ldots, p_7)$ *avec* $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$, $p_2 = p_3 = \frac{1}{16}$, $p_4 = p_5 = p_6 = p_7 = \frac{1}{32}$. *Trouver le code de Shannon associé.*

Exercice. *Notre source sans mémoire qui produit les huit lettres A, B, C, D, E, F, G, H selon la distribution de probabilité* $\mathbf{p} = (p(A), p(B), \ldots, p(H))$ *, avec* $p(A) = \frac{27}{64}$, $p(B) = p(C) = \frac{3}{16}$, $p(D) = \frac{1}{16}$, $p(E) = p(F) = \frac{3}{64}$, $p(G) = \frac{1}{32}$, $p(H) = \frac{1}{64}$.

1. *Trouver le code de Shannon associé.*
2. *Calculer la longueur moyenne* $\bar{l}$ *des mots code.*

5.3 Coder plusieurs symboles successifs en même temps

Pour le codage Huffman nous avons déjà vu que le codage par blocs de n symboles permet de prendre en compte plusieurs symboles en même temps. C'est un codage vectoriel (en dimension n) qui remplace la source initiale par une "extension d'ordre n" ayant comme symboles des vecteurs de n symboles successifs de la source initiale.

Le théorème de Shannon pour le codage de source sans pertes propose de réaliser un codage vectoriel en grande dimension pour toute source. pour s'approcher de l'entropie. Pour coder une source discrète sans mémoire sur un alphabet à N symboles en dimension n l'alphabet de l'"extension d'ordre n" augmente exponentiellement avec n. Donc, le codage vectoriel en grande dimension est d'un intérêt purement théorique.

Pour résoudre ces problems d'autres systèmes de codage de source sans pertes ont été proposés. Ils permettent aussi de prendre en compte les dépendances temporelles (d'un symbole à l'autre) de la source (avec mémoire). Ces systèmes de codage permettent de coder une source quelconque sans connaître a priori ses statistiques (c'est ce qu'on appelle du codage "universel"). Les plus connus sont les systèmes de codage de Lempel-Ziv (1976)et de codage arithmétique (1982). Le codage arithmétique est probablement le plus important dans les applications et les normes actuelles. Il est une extension itérative d'une technique de codage connue depuis les années 50, appelée codage d'Elias (ou de Shannon-Fano-Elias).

5.4 Le codeur d'Elias

Le codeur d'Elias était, à l'origine, considéré comme une construction purement académique. Sa première présentation (tardive) date de 1968. C'est entre 1976 et 1979 (Pasco, Jones, Rubin) que l'on découvrit son intérêt pratique de sa variante **binaire** $\longrightarrow$ **binaire**. Le codeur d'Elias code optimalement, c.àd. il approche l'entropie de la source :

$$H(\mathbf{p}) \leq \frac{l}{n} < H(\mathbf{p}) + \frac{1}{n}$$

Soit une source binaire (sans mémoire), produisant selon la distribution de probabilité $\mathbf{p} = (p_0, p_1)$, avec $p_0 \geq p_1$. Le codeur d'Elias construit un arbre binaire d'intervalles, obtenu par p_0-dichotomie. Soit les intervalles $[A_k, B_k[$ les intervalles $[A_{k+1}, B_{k+1}[$ sont calculés :

$$D \;=\; A_k \;+\; p_0 \cdot (B_k - A_k)$$

$$A_{k+1} = A_k, B_{k+1} = D \text{ (codé avec 0) et } A_{k+1} = D, B_{k+1} = B_m \text{ (codé avec 1)}$$

Les n premiers bits $a_{j_1} a_{j_2} \ldots a_{j_n}$ d'un flot source binaire sont interprété comme **chemin** dans cet arbre. Le chemin pointe sur un intervalle $[A, B[$. Le mot **code d'Elias** $c(a_{j_1} a_{j_2} \ldots a_{j_n})$ est le mot code de l'intervalle $[A, B[$, c.a.d. le développement dyadique A jusqu'au $l = \lceil \log_2 \frac{1}{B-A} \rceil$ "décalage de la virgule "

Exemple. *Soit une source binaire (sans mémoire) produisant selon la distribution de probabilité* $\mathbf{p}$ *donnée par* $p_0 = \frac{3}{4}, \quad p_1 = \frac{1}{4}$.

Trouvez le mot code d'Elias de la sequence 001 produite par la source, c.a.d. de l'intervalle $[\frac{27}{64}, \frac{9}{16}[$

Premier pas : Pour le codage, il faut trouver dans l'hierarchie de l'arbre de p_0 dichotomie le chemin ("poupée russe") d' intervalles **emboîtés** qui correspond à la sequence s produite par la source. Calcul de $D = \frac{3}{4} = 0.11$

$$s_1 = 0 \text{ et donc } A_1 = 0, B_1 = \frac{3}{4}$$

Calcul de $D = \frac{9}{16} = 0.1001$

$$s_1 s_2 = 00 \text{ et donc } A_2 = 0, B_2 = \frac{9}{16}$$

Calcul de $D = \frac{27}{64} = 0.011011$.

$$s_1 s_2 s_3 = 001 \text{ et donc } A_3 = \frac{27}{64}, B_3 = \frac{9}{16}$$

Deuxième pas : Determiner la precision du développement dyadique. La longueur du codage est donnée par l'information qui correspond à l'interval $B - A = \frac{9}{16} - \frac{27}{64} = \frac{9}{64}$. Donc $\log_2 \frac{64}{9} < \log_2 \frac{64}{8} = 3$.

$$\frac{27}{64} = 0.\alpha_1\alpha_2\alpha_3\cdots$$

Troisième pas : Obtenir le développement binaire du nombre $A = \frac{27}{64}$ qui est entre 0 et 1.

$$2 \times \frac{27}{64} = \alpha_1.\alpha_2\alpha_3\cdots$$

$$2 \times \frac{27}{64} = \frac{27}{32} < 1 \Longrightarrow \alpha_1 = 0 \text{ et } \frac{27}{32} = 0.\alpha_2\alpha_3\cdots$$

$$2 \times \frac{27}{32} = \alpha_2.\alpha_3\cdots$$

$$2 \times \frac{27}{32} = \frac{27}{16} = 1 + \frac{11}{16} \geq 1 \Longrightarrow \alpha_2 = 1 \text{ et } \frac{11}{16} = 0.\alpha_3\cdots$$

$$2 \times \frac{11}{16} = \alpha_3$$

$$2 \times \frac{22}{16} > 1 \Longrightarrow \alpha_3 = 1 \Longrightarrow$$

$$[\frac{27}{64}, \frac{9}{16}[= \alpha_1\alpha_2\alpha_3\cdots = 011$$

Exemple. *Soit une source binaire (sans mémoire) produisant selon la distribution de probabilité* **p** *donnée par* $p_0 = \frac{3}{4}$, $p_1 = \frac{1}{4}$.

Décodez le mot code d'Elias : 1000000.

Notre intervalle est $[A_n, B_n[$ avec $A_n = 0.1000000*$ et $\frac{1}{2^7} \leq (B_n - A_n) < \frac{1}{2^6}$.

Pour le décodage, il faut retracer les décisions du **codeur** dans l'hierarchie de l'arbre de p_0 dichotomie : un codage est décrit par une "poupée russe" d' intervalles **emboîtés**.

Premier pas : Calcul de $D = \frac{3}{4} = 0.11$

$$A_n < D \Longrightarrow [A_n, B_n[\subset [0, D[\equiv \text{ l'intervalle de 0.}$$

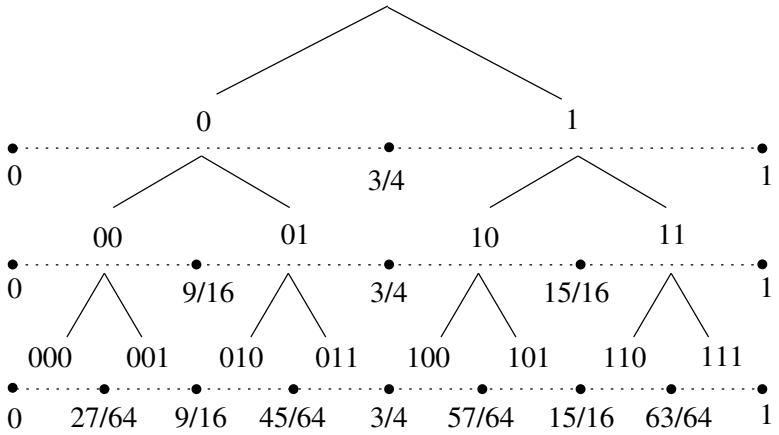

FIGURE 5.2 – Arbre p_0 dichotomique

$\Longrightarrow \quad s_1 = 0$ et $A_1 = 0, B_1 = \frac{3}{4}$.

Deuxième pas : Calcul de $D = \frac{9}{16} = 0.1001$

$$A_n < D \Longrightarrow [A_n, B_n[\subset [0, D[\equiv \text{ l'intervalle de } \quad 00.$$

$\Longrightarrow s_1 s_2 = 00$ et $A_2 = 0, B_2 = \frac{9}{16}$.

Troisième pas : Calcul de $D = \frac{27}{64} = 0.011011$.

$$D < A_n \Longrightarrow [A_n, B_n[\subset [D, B_2[\equiv \text{ l'intervalle de } 001.$$

$\Longrightarrow \quad s_1 s_2 s_3 = 001$ et $A_3 = \frac{27}{64}, B_3 = \frac{9}{16}$.

Quatrième pas : Calcul de$D = \frac{135}{256} = 0.10000111$

$$A_n < D \Longrightarrow [A_n, B_n[\subset [A_3, D[\text{ l'intervalle de } 0010.$$

$\Longrightarrow \quad s_1 s_2 s_3 s_4 = 0010$ et $A_4 = \frac{27}{64}, \qquad B_4 = \frac{135}{256}$.

Cinquième pas : Calcul de $D = \frac{513}{1024} = 0.1000000001$. A_n et D ont le **même** développement binaire - jusqu'à la partie masquée de A_n.

Pour un décodage cohérent et logique d'un code préfixe on considere $A_n = D$ et $s_5 = 1$ (c.àd. $[A_n, B_n[\subset [D, B_4[$). Alors on a trouvé un flot source

$$\mathbf{s} = s_1 s_2 s_3 s_4 s_5 = 00101$$

avec le mot code 100000 (noter que $I(00101) = 3 \times 0.42 + 2 \times 2 = 5.26$).

Remarque. *Si on complete* $\mathbf{s} = s_1 s_2 s_3 s_4 s_5$ *avec des "0",* $\frac{513}{1024}$ *restera alors toujours la borne inférieure* $A_5 = A_6 = A_7$ *etc. de l'intervalle code. On peut avoir ainsi d'autres suites avec le même mot code :*

* $s_1 s_2 s_3 s_4 s_5 s_6 s_7 = 0010100$ *(avec* $I(0010100) = 5 \times 0.42 + 2 \times 2 = 6.1$*)*
* $s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 = 00101000$ *(avec* $I(s_1 \dots s_8) = 6.52$ *)*
* $s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 = 001010000$ *(avec* $I(s_1 \dots s_9) = 6.94$ *)*

Elles doivent être préfixe une de l'autre (exercice !).

Noter qu'en pratique on connaît la longueur du flot source.

5.5 Codage arithmétique

Le premier avantage du codage arithmétique est que chaque caractère peut être codé sur un nombre non-entier de bits. L'algorithme ne code pas les fichiers caractère par caractère mais par chaînes de caractères, plus ou moins longues suivant la capacité de la machine à coder des réels plus ou moins grands.

Un autre avantage du codage arithmétique est qu'il est un codage adaptatif pour les cas general d'une source avec mémoire. Plutôt que de supposer connue une fois pour toutes la distribution de probabilité conjointe de la source $p(x_0, ..., x_M)$ (par modélisation et/ou estimation sur toute la source), il est possible d'estimer au fur et à mesure les probabilités conditionnelles $p(x_M | x_0, ..., x_M)$.

Le codage arithmétique est optimal dans le cas général des sources avec mémoire.

Proposition. *Soit une source avec/sans mémoire, produisant* N *symboles* $a_0, a_1, \dots, a_{N-1}$ *selon la distribution de probabilité* $\mathbf{p} = (\ p_0, p_1, \dots, p_{N-1}\ )$*. Soit* C *le codage arithméthique, alors :*

$$H(\mathbf{p}) \leq \bar{l} < H(\mathbf{p}) + \frac{2}{n} \tag{5.5.1}$$

Soit une sequence à coder $x_0, \cdots, x_M$ de M symboles. La distribution de probabilité des symboles de source vérifie la relation :

$$\sum_{i=0}^{N-1} p_i = 1$$

On peut donc construire une partition (un "découpage") du segment $[0, 1[$ (de longueur 1) en intervalles contigus $I_0, I_1, \dots, I_M$, où chaque I_i est de longueur $p(x_i)$. L'idée du codage arithmétique est d'itérer la procédure d'Elias au fur et à mesure du temps, afin coder plusieurs symboles de source successifs. Ainsi on peut prendre en compte même les corrélations éventuelles des symboles de la source (avec mémoire).

On commence par appliquer la procédure d'Elias pour le symbole x_0 du segment de flot $x_0x_1x_2\cdots x_M$ on obtient une représentation codée dans un intervalle I_0 de longueur $p(x_0)$. Pour tenir compte de x_0, on considère non pas les probabilités $p_(x_1)$, mais les probabilitÃ©s conditionnelles $p(x_1|x_0)$ qui vérifient la relation :

$$\sum_{j=0}^{N} p(j|x_0) = 1$$

On recommence alors la procédure d'Elias, non plus sur le segment [0,1[, mais sur l'intervalle I_0 qu'on re-partitionne en intervalles I_1 plus fins, de longueurs proportionnelles aux $p(x_1|x_0)$ et ainsi de suite.

Dans le cas d'une source sans mémoire on re-partitionne en intervalles pareil que la première fois.

Après n itérations de l'algorithme d'Elias le résultat du codage est donc formé des décimales du nombre $c \in I_{N-1}$ avec une précision suffisante pour qu'il appartienne à un intervalle I_{N-1} unique.

En théorie, il suffit de disposer d'une implantation d'une arithmétique à précision infinie (ou suffisamment grande). Ceci explique le terme "codage arithmétique".

En pratique, il n'y a pas une implantation d'une arithmétique à précision infinie. C'est la raison pour laquelle il est nécessaire de procéder à des "remises à l'échelle" chaque fois que la précision machine va être atteinte pour éviter les problèmes d'underflow. Ces remises à l'échelle sont re-parcourues lors du décodage.

Le décodage procède dans le même sens que le codage : on regarde d'abord dans quel intervalle I_0 se trouve c, on en déduit x_0, puis on regarde à l'intérieur de I_0 dans quel intervalle I_1 se trouve c, on en déduit x_1, et ainsi de suite.

Algorithme de codage

1. On initialise un premier intervalle avec deux bornes : la borne inférieure Lc = 0 et la borne supérieure Hc = 1 (correspondant à la probabilité de choisir un premier symbole x_0 parmi tous les symboles de la source). La taille de l'intervalle : taille= $H_c - L_c$.
2. Cet intervalle est partitionné en N sous-intervalles $[L_k, H_k[$ en fonction des probabilités de chaque symbole a_k de la source. Ce sont les partitions initiales avec longueur $L_k - H_k = p_k$. Nous avons :

$$L_k = L_c + taille \times \sum_{i=1}^{k-1} p(x_i) \text{ et } H_k = L_c + taille \times \sum_{i=1}^{k} p(x_i)$$

3. On choisit le sous-intervalle correspondant au prochain symbole x_k qui apparaît dans la séquence. On redéfinit alors l'intervalle initial [Lc, Hc[:

$$L_c = L_c + taille \times L_k \text{ et } H_c = L_c + taille \times H_k$$

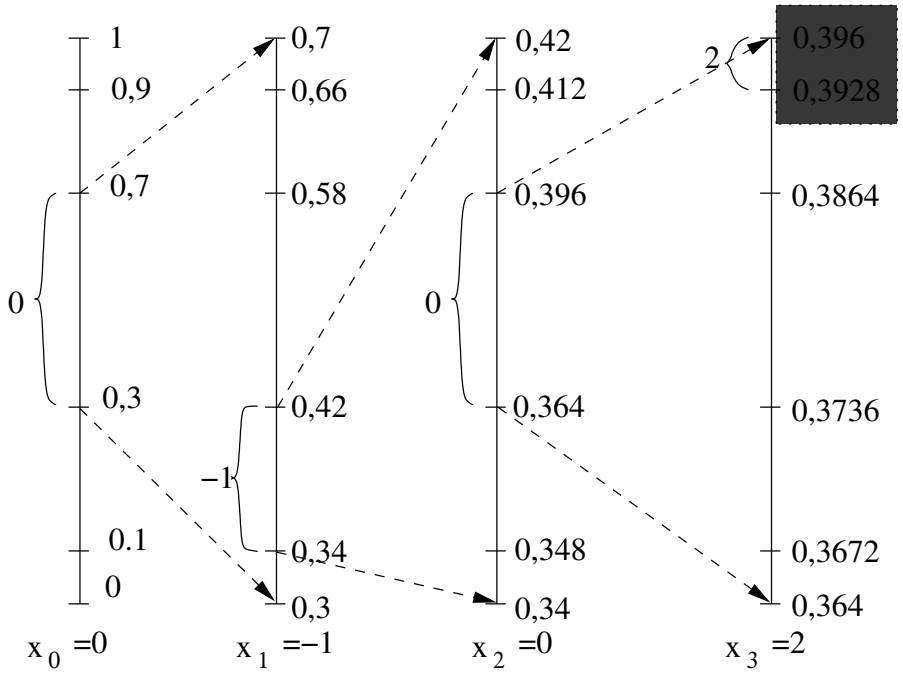

FIGURE 5.3 – Subdivision Codage Arithmétique

Exemple. *Soit la source discrète sur l'alphabet -2, -1, 0, 1, 2 qui définit une série de 5 vecteurs de mouvements verticaux possibles (vecteurs utilisés en codage vidéo). La probabilité de production de la source est* $\mathbf{p} = (0.1, 0.2, 0.4, 0.2, 0.1)$.

Coder la séquence de vecteurs de mouvement (0, -1, 0, 2)

Pour coder 0, -1, 0, 2, on divise l'intervalle [0, 1[en 5 partitions initiales, puis on se place sur le sous-intervalle correspondant au premier vecteur de mouvement de la séquence (de valeur 0), c.à.d. [0.3, 0.7[. taille(1)=0.4

Pour le prochain symbole de la séquence, le vecteur de mouvement 1, on subdivise la partition initiale du premier vecteur 0 en autant d'intervalles qu'il y'a de vecteurs possibles. On procède récursivement pour tous les symboles (les vecteurs de mouvement) de la séquence. Nous avons taille(2)=0.08, taille(3)=0.032, taille(4)=0.0032. Dès lors, on peut coder la séquence (0, -1, 0, 2) par tout nombre rééel appartenant à l'intervalle [0,3928 ; 0,396[. Le nombre 0,3945 code cette séquence. On le code avec 8 bits :

$$0.3945 = 0 \times 2^{-1} + 2^{-2} + 2^{-3} + 0 \times 2^{-4} + 0 \times 2^{-5} + 2^{-6} + 0 \times 2^{-7} + 2^{-8}$$

Nous utilisons donc $8/5 = 1{,}6$ bits/symbole.

Algorithme de décodage

1. Initialiser un premier intervalle avec deux bornes : la borne inférieure Lc = 0 et la borne supérieure Hc = 1. La taille de l'intervalle : taille= $H_c - L_c$.
2. Trouver le sous-intervalle $[L_k, H_k[$ du symbole, tel que :

$$L_k \leq \frac{(MotCode - L_c)}{taille} < H_k$$

3. Obtention du symbole : x_k
4. Mettre à jour le sous-intervalle :

$$L_c = L_c + taille \times L_k \text{ et } H_c = L_c + taille \times H_k$$

5. Répéter les étapes 2, 3, 4 et 5 jusqu'à obtenir le décodage de tous les symboles de la séquence.

Exemple. *Soit la source discrète sur l'alphabet -2, -1, 0, 1, 2 qui définit une série de 5 vecteurs de mouvements verticaux possibles (vecteurs utilisés en codage vidéo). La probabilité de production de la source est* $\mathbf{p} = (0.1, 0.2, 0.4, 0.2, 0.1)$.

Trouver la séquence de vecteurs de mouvement qui correspond au mot-code $M = 0.3945$.