École Internationale des Sciences du Traitement de l'Information

Département "Informatique"

Introduction à la Théorie de l'Information

Notes de cours

Anya Désilles

Année scolaire 2009/2010

Table des matières

Bibliographie

[1] Robert B. Ash. *Information theory.* Dover Publications, Inc, 1965.

[2] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003. Disponible pour téléchargement gratuit sur http ://www.inference.phy.cam.ac.uk/mackay/itila/book.html.

[3] John R. Pierce. *An Introduction to Information Theory. Symbols, Signals and noise.* Dover Publications, Inc, New York, 1980.

[4] Fazlollah M. Reza. *An introduction to information theory.* Dover Publications, Inc, 1994.

Introduction

La théorie de l'information trouve ses origines dans les débuts des communications électriques. Le premier télégraphe élaboré par S. Morse entre 1832 et 1838 a permis de communiquer n'importe quel texte à l'aide de signaux électriques. Dans le code de Morse chaque lettre de l'alphabet est représentée par une séquence de

- **points** (courant électrique de courte durée) ;
- **traits** (courant de longue durée) ;
- **espaces** (absence de courant) ;

On peut remarquer que dans la version définitive du code la lettre "e", la plus fréquente dans l'anglais, est représentée par la séquence la plus courte : un seul point. À cette époque, S. Morse n'avait pas fait d'analyse théorique pour arriver à cette conclusion, mais plutôt des observations empiriques. Son but en effet était de concevoir le code tel que la saisie d'un texte par un opérateur soit, en moyenne, la plus rapide possible. Aujourd'hui, la théorie moderne de communication a montré qu'il est possible de gagner à peine 15 pour cent de temps de saisie par rapport au code de Morse.

Quelques années plus tard, une première ligne télégraphique est installée entre Washington et Baltimore. En enterrant les câbles, S. Morse rencontre une nouvelle difficulté : le milieu dans lequel ces derniers se trouvent influe sur la qualité de transmission. En particulier, il remarque que si l'opérateur saisit trop vite son code, le signal reçu est indéchiffrable. Les points, les traits, les espaces sont confondus dans un seul signal d'intensité moyenne. Ainsi apparaît le problème de perturbations dues aux conditions physiques de transmission de l'information et, comme condition de bonne réussite, la nécessité de limiter le débit d'émission de symboles.

Bien plus tard, dans les années 1940, l'ingénieur et mathématicien C. Shannon et le mathématicien Warren Weaver, ont formalisé le processus de communication et donné les outils mathématiques permettant de répondre aux questions posées empiriquement par les premiers télégraphes. Ils proposent en particulier, une représentation schématique de processus de communication, connue sous le nom de paradigme de Shannon.

Ce modèle est une **approximation linéaire de processus de communication** qui met l'accent sur les aspects purement techniques de transmission d'un message. On peut résumer ce modèle de façon suivante :

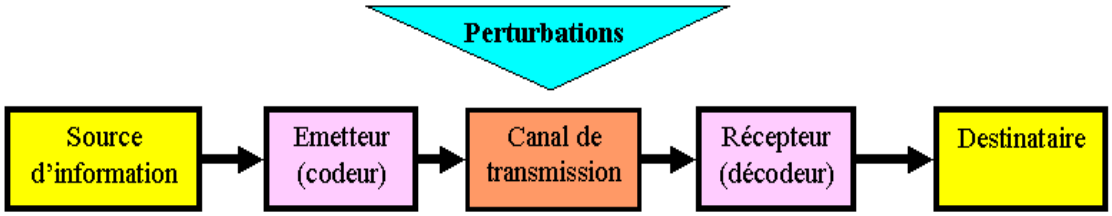

FIGURE 1 – Paradigme de Shannon

1. la source d'information choisit un message M parmi un certain nombre de messages possibles ;
2. l'émetteur transforme le message en signal S compatible physiquement avec le mode de transmission choisi. On dit qu'il **encode** le message ;
3. le signal S est alors soumis à l'entrée d'un canal de transmission ;
4. lors de la transmission des perturbations peuvent intervenir et transformer le signal envoyé ; on parle alors **de bruit de canal** B ;
5. à la sortie du canal, le signal $\tilde{S}$ éventuellement entaché d'erreurs dues au bruit, est soumis au décodeur qui le transforme en message $\tilde{M}$ lisible par le destinataire.

Voici un exemple pour illustrer ce schéma. Imaginez qu'un ami vous envoie une carte postale du lieu de ses vacances. La carte voyage sans incident jusqu'à la ville où vous habitez. Le jour où votre facteur doit enfin vous l'apporter, il la fait tomber par mégarde. Malheur ! Il pleut. L'enveloppe a pris l'eau. C'est ainsi que vous retrouvez les quelques lignes écrites par votre ami à moitié illisibles. Allez vous pouvoir reconstituer le contenu complet du message ?

Dans cet exemple, **la source d'information** est votre ami, et plus précisément, son cerveau. C'est encore lui qui joue ici le rôle de **l'émetteur**, en transformant ses idées en mots et ensuite en écrivant les mots à l'aide de lettres de l'alphabet. **Le canal de transmission** est ici représenté par les services postaux. Les dégâts causés par l'eau à la carte postale représentent **le bruit de canal**. Les questions que l'on pourrait poser dans cette situation sont les suivantes :

1. Y a-t-il un moyen de préparer le message de façon à éviter les désagréments causés par les erreurs de transmission et garantir à l'arrivée la lisibilité du message ? Par exemple, pourrait-on écrire avec un encre indélébile ? **C'est le problème de codage de source**.
2. Y a-t-il un moyen de transporter le signal dans le canal de manière à limiter les perturbations ? Par exemple, protéger le courrier dans des enveloppes imperméables ? **C'est le problème de codage de canal.**
3. Comment évaluer l'incertitude que le récepteur a sur le contenu réel du message reçu ? **C'est le problème de mesure de l'information.**

Ce modèle, certes très simplifié, de la communication a servi de base dans le développement de la théorie de l'information à partir des années 1940. Il s'inscrit dans une description plus générale, donnée par Warren Weaver des trois niveaux de problèmes de communication :

Niveau 1 : Technique Avec quelle précision peut on transmettre les symboles de la communication ?

Niveau 2 : Sémantique Dans quelle mesure les symboles véhiculent la signification ?

Niveau 3 : Efficacité Dans quelle mesure la signification reçue influence le comportement et l'action du destinataire ?

La théorie de l'information s'intéresse uniquement aux problèmes du premier niveau de communication. En particulier, le sens des messages traités n'a aucune importance. L'information quantifiable associée à un message donné n'est pas dans son sens sémantique mais dans sa rareté. Un message, même très significatif, connu à l'avance par le récepteur ne lui apporte aucune information au sens technique du terme. Par contre, la réception d'un message très improbable mais démuni de tout sens est dans ce cadre considérée comme événement porteur d'une grande quantité d'information.

Les chapitres qui suivent ont pour objectif d'introduire les concepts principaux de la théorie de l'information et les théorèmes fondamentaux. Ces derniers établissant les limites théoriques en matière de codage de l'information et de transmission. Enfin, nous consacrerons la deuxième moitié du cours à l'étude de quelques algorithmes de base de codage de source.

Chapitre 1

Modélisation mathématique d'une source d'information

1 Représentation mathématique d'une source

Soit une source (un émetteur) produisant des symboles d'un alphabet fini $\Omega = \{\omega_1, \ldots, \omega_m\}$. Nous supposons que chaque symbole $\omega_i,\ i = 1, 2, \ldots, m$ est émis aléatoirement avec une probabilité connue p_i de telle sorte que $\sum_{i=1}^{m} p_i = 1$.

On considère alors l'expérience aléatoire consistant à observer un symbole émis. À cette expérience nous associons une variable aléatoire, notée X à valeurs dans Ω. C'est une variable aléatoire discrète, dont la loi est définie par l'ensemble des probabilités d'émission des symboles de l'alphabet :

$$P[X = \omega_i] = p(\omega_i) = p_i,\ i = 1, \ldots, m$$

Exemple 1.1. Soit X la variable associée à une source binaire, produisant des symboles 0 et 1 avec les probabilités respectives p et $q = 1 - p$. Nous avons ici l'alphabet constitué de deux symboles $\Omega = \{0, 1\}$ et $P[X = 0] = p, \quad P[X = 1] = q$.

Exemple 1.2. Soit Y la variable associée à une source disposant d'un vocabulaire de 20 mots, $\Omega = \{$le, la, pomme, fruit, est, un, arbre, pommier, jardin, ombre, fait, soleil, vent, pluie, grandit, quand, ne, pas, mais, oiseau$\}$ et que tous les mots sont équiprobables. Ainsi, n'importe quel mot $M_i \in \Omega$ du vocabulaire est choisi avec la probabilité $\frac{1}{20}$ et $P[X = M_i] = \frac{1}{20},\ i = 1, \ldots, 20$.

Dans la suite nous allons considérer des événements associées à la variable aléatoire X comme des sous-ensembles de Ω et la mesure de probabilité définie pour $A \subset \Omega$ par

$$P[A] = P[X \in A].$$

Si on considère l'émission de plusieurs symboles successifs, à des instants de temps précis $\{t_1, t_2, \ldots, t_n\}$ on peut y associer la suite de variables aléatoires $\{X_1, X_2, \ldots, X_n\}$, chacune correspondant à l'émission d'un symbole $s(t) \in \Omega,\ t \in \{t_1, t_2, \ldots, t_n\}$. Nous supposons pour l'instant que tous les symboles successifs sont émis de façon indépendante les uns des autres et avec les mêmes probabilités d'émissions : $p_i = P[s(t) = \omega_i]$. Cela signifie que les variables aléatoires $X_k,\ k = 1, \ldots, n$ sont **deux à deux indépendantes et identiquement distribuées**. On parle alors d'une source stationnaire et sans mémoire. Nous allons préciser la signification formelle de ces termes plus loin.

2 Information propre

Pour introduire l'ensemble des axiomes qui définissent l'information propre d'un événement $A \subset \Omega$, nous allons nous servir du premier exemple de source binaire ci-dessus (voir 1.1). Soit donc une source binaire d'alphabet $\Omega = \{0, 1\}$ et la variable aléatoire X associée avec les probabilités données :

$$P(0) = P[X = 0] = p \text{ et } P(1) = P[X = 1] = q = 1 - p$$

Dans ce qui suit, nous allons construire de façon axiomatique, une fonction

$$h : \mathcal{P}(\Omega) \to \mathbb{R}$$

associant à un événement $A \subset \Omega$ la mesure de la quantité d'information contenue dans A. Nous l'appellerons **"information propre d'un événement** A**"**.

Dans la discussion du chapitre 1 nous avons déjà indiqué de façon intuitive que la quantité d'information apportée par un événement devrait être d'autant plus grande que l'événement est rare, improbable. Nous pouvons donc postuler que la quantité d'information d'un événement doit être de la forme :

$$h(A) = f\left(\frac{1}{P(A)}\right)$$

où f est une fonction croissante.

En particulier, dans notre exemple de source binaire $A = \{\omega\}$, avec $\omega \in \Omega$ on doit avoir

$$h(\omega) = f\left(\frac{1}{P(\omega)}\right),\ \omega \in \{0, 1\}$$

Si un événement A est certain, c'est-à-dire si $P[A] = 1$, il n'apporte aucune information. La fonction f que l'on recherche doit donc vérifier la condition limite suivante :

$$\lim_{x \to 1} f(x) = 0$$

Par ailleurs, il serait logique de s'attendre à ce que l'information apportée par deux événements indépendants, A et B, soit égale à la somme des informations apportées par chacun de ces événements :

$$h(A \cap B) = h(A) + h(B)$$

et donc

$$h(A \cap B) = f\left(\frac{1}{P(A \cap B)}\right) = f\left(\frac{1}{P(A)} \cdot \frac{1}{P(B)}\right) = f\left(\frac{1}{P(A)}\right) + f\left(\frac{1}{P(B)}\right)$$

Ainsi, la fonction f doit vérifier la propriété suivante :

$$f(xy) = f(x) + f(y)$$

L'unique fonction (à une constante multiplicative près) qui vérifie toutes ces propriétés est la fonction logarithme. Ainsi, nous définissons l'information propre d'un événement de façon suivante.

Définition 1.1 (Information propre)**.** Soient $\Omega = \{\omega_1, \ldots, \omega_m\}$ un alphabet discret et X la variable aléatoire associée. Pour tout événement $A \subset \Omega$ la quantité d'information propre de A est définie par

$$h(A) = -\log_2(P(A)).$$

Remarque 1.1. Dans la suite nous allons considérer le logarithme de base 2. L'unité de mesure de l'information est alors le *Shannon*. Le changement de base du logarithme provoque la modification de la constante multiplicative car

$$\log_a(x) = \log_a(b) \cdot \log_b(x)$$

Voici quelques propriétés élémentaires de la fonction $h(A)$ qui sont les conséquences immédiates des propriétés de la fonction logarithme.

Proposition 1.1 (Propriétés de l'information propre)**.**
1. *La quantité d'information est toujours une grandeur positive :* $\forall A \subset \Omega,\ h(A) \geq 0$
2. *Elle est nulle si et seulement si l'événement A est certain :* $h(A) = 0 \Leftrightarrow P(A) = 1$.
3. $\lim\limits_{P(A)\to 0} h(A) = +\infty$.

3 Information conditionnelle

Nous avons déjà évoqué dans la section précédente la question d'évaluation de la quantité d'information apportée par la réalisation conjointe de deux événements A et B. Nous avons postulé que dans le cas de deux événements A et B indépendants on a

$$h(A \cap B) = h(A) + h(B).$$

Or, dans le cas où A et B ne sont pas indépendants, on peut utiliser la notion de probabilité conditionnelle définie par (voir le cours "Probabilités" de Marietta Manolessou) :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

d'où les relations :

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B).$$

Alors, la quantité d'information propre de $A \cap B$ est :

$$h(A \cap B) = -\log_2(P(A)P(B|A)) = -\log_2(P(A)) - \log_2(P(B|A)) = h(A) - \log_2(P(B|A)).$$

Définition 1.2 (Information conditionnelle)**.** On appelle information conditionnelle de B sachant A la quantité

$$h(B|A) = -\log_2(P(B|A)).$$

Remarque 1.2. On peut donc écrire $h(A \cap B)$ de la façon suivante :

$$h(A \cap B) = h(A) + h(B|A).$$

On peut interpréter la quantité d'information conditionnelle comme suit : $h(B|A)$ représente l'information apportée par l'observation de l'événement B *qui n'a pas déjà été* apportée par l'observation de A.

4 Information mutuelle

Définition 1.3 (Information mutuelle)**.** On appelle information mutuelle de deux événements B et A la quantité

$$i(A;B) = h(A) - h(A|B) = \log_2 \frac{P(A|B)}{P(A)} = \log_2 \frac{P(A \cap B)}{P(A)P(B)}.$$

5 Information *vs* incertitude

La définition de l'information associée à un événement A implique que cette quantité est d'autant plus grande que l'événement est rare (c'est-à-dire que sa probabilité est proche de zéro). Ainsi, on peut assimiler l'information apportée par l'observation de A et l'incertitude que nous avons sur sa réalisation, ou encore la difficulté de prévoir A.

Considérons par exemple l'expérience de lancé d'un dé équilibré à 6 faces. Soit X la variable aléatoire associée :

$$\forall \omega \in \{1,2,3,4,5,6\}, \quad P[X = \omega] = \frac{1}{6}.$$

Imaginons que le résultat soit $X = 2$. Soit A l'événement associé. On a $P(A) = \frac{1}{6}$.

Si on essaye de deviner ce résultat, on a une chance sur 6 de trouver la réponse correcte. L'information associée est

$$h(A) = -\log_2\left(\frac{1}{6}\right) = \log_2(6) \simeq 2.585$$

Imaginons que l'on affirme que le résultat du lancé est un nombre pair. Soit B l'événement associé. On a $P(B) = \frac{1}{2}$. Alors, la quantité d'information que nous apporte ce renseignement est

$$h(B) = -\log_2\left(\frac{1}{2}\right) = \log_2(2) = 1$$

Cette information, augmente notre chance de réussite. En effet, nous avons maintenant la probabilité $P(A|B) = \frac{1}{3}$ de deviner le résultat. Et nous retrouvons la décomposition de l'information mutuelle associée à $A \cap B$ en somme de l'information apportée par l'observation de B et l'information conditionnelle $h(A|B)$:

$$h(A \cap B) = h(B) + h(A|B) = -\log_2\left(\frac{1}{2}\right) - \log_2\left(\frac{1}{3}\right) = \log_2(2) + \log_2(3) = \log_2(6).$$

6 Entropie d'une source

On considère maintenant une source stationnaire et sans mémoire, produisant à des instants de temps précis $\{t_1, t_2, \ldots, t_k\}$ des symboles $s(t) \in \Omega,\ t \in \{t_1, t_2, \ldots, t_k\}$ d'un alphabet Ω donné.

L'entropie d'une telle source représente la quantité moyenne d'information propre associée à l'observation de chacun des symboles possibles.

Supposons que l'alphabet de la source $\Omega = \{s_i,\ i = 1, \ldots, n\}$ comporte n symboles et que la variable aléatoire associée à l'observation d'un symbole émis, X, ait la distribution de probabilité donnée $\{p_i = P[X = s_i],\ i = 1, \ldots, n\}$. L'entropie de la source est alors la fonction des n probabilités $H(p_1, p_2, \ldots, p_n)$.

Tout comme pour l'information propre, l'entropie peut être définie par l'ensemble d'axiomes suivant :

1. Si tous les symboles sont équiprobables l'entropie $H(1/n, \ldots, 1/n) = f(n)$ est une fonction de n, la taille de l'alphabet. Nous souhaitons qu'elle soit la mesure de l'incertitude associée à la source, ou encore de la difficulté à prédire le symbole émis. Il est alors naturel de supposer que l'entropie augmente avec la taille de l'alphabet. Par exemple, il est plus difficile de deviner le résultat de lancé d'un dé à 6 faces que celui d'une pièce de monnaie. Ainsi, la fonction $f(n) = H(1/n, \ldots, 1/n)$ ci-dessus doit être **croissante.**
2. Considérons une source qui émet des couples de symboles puisés chacun dans un alphabet indépendant de tailles respectives l et m. La taille de l'alphabet produit est alors $l \cdot m$. Or, l'observation de chaque couple apporte la somme d'informations liées à chaque symbole. Ainsi, la fonction f doit vérifier la **propriété d'additivité** $f(l \cdot m) = f(l) + f(m)$.
3. La fonction $H(p_1, \ldots, p_n)$ **est continue** selon chacune de ses variables.

4. **Propriété de groupes.** Si l'on divise l'alphabet donné Ω en deux parties disjointes, $\Omega_1 = \{s_1, \ldots, s_r\}$ et $\Omega_2 = \{s_{r+1}, \ldots, s_n\}$ chacune peut être considérée comme alphabet. Notons $Q_1 = P(\Omega_1) = \sum_{k=1}^{r} p_k$ et $Q_2 = P(\Omega_2) = \sum_{k=r+1}^{n} p_k$. On a $Q_1 + Q_2 = 1$. On peut alors définir sur Ω_1 et sur Ω_2 les distributions des probabilités respectives $\left\{\frac{p_1}{Q_1}, \ldots, \frac{p_r}{Q_1}\right\}$ et $\left\{\frac{p_{r+1}}{Q_2}, \ldots, \frac{p_n}{Q_2}\right\}$. Alors l'entropie associée à l'alphabet Ω se décompose de la façon suivante :

$$H(p_1, \ldots, p_r, p_{r+1}, \ldots, p_n) = H(Q_1, Q_2) + Q_1 H\left(\frac{p_1}{Q_1}, \ldots, \frac{p_r}{Q_1}\right) + Q_2\left(\frac{p_{r+1}}{Q_2}, \ldots, \frac{p_n}{Q_2}\right)$$

5. **Symétrie.** Il est naturel de supposer que la quantité moyenne d'information associée à un alphabet Ω ne dépend pas de l'ordre de numérotation des symboles. Ainsi, la permutation de deux arguments de la fonction d'entropie ne la modifie pas :

$$H(p_1, \ldots, p_i, \ldots, p_j, \ldots, p_n) = H(p_1, \ldots, p_j, \ldots, p_i, \ldots, p_n)$$

La fonction vérifiant ces quatre axiomes est une unique à une constate multiplicative près. Après la publication en 1948 par C. Shannon de son article "The Mathematical Theory of Communication", de nombreux mathématiciens ont travaillé l'élaboration de la définition axiomatique de l'entropie. On peut citer Khinchin (1957), Fadeev(1956), Kullback 1958, Rényi 1962. En sélectionnant des ensembles d'axiomes différents, tous ont abouti à la même fonction. D'où la définition qui suit.

Définition 1.4 (Entropie d'une source)**.** Soient $\Omega = \{\omega_1, \ldots, \omega_m\}$ l'alphabet fini d'une source et X la variable aléatoire associée t.q. $P[\omega_i] = p_i, \quad i = 1, \ldots, m$. On appelle **entropie** ou encore **quantité moyenne d'information** de la source la quantité

$$H(X) = H(p_1, p_2, \ldots, p_n) = E[h(\omega)] = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

L'unité de mesure de cette quantité est le "bit par symbole".

L'entropie représente la quantité d'information que l'on obtient en moyenne, en observant les symboles en sortie de la source pendant suffisamment longtemps ou encore, la valeur de l'information moyenne obtenue en observant simultanément les symboles en sortie d'un grand nombre de sources identiques.

Elle mesure également le nombre de bits moyens nécessaires pour coder chaque symbole de la source. C'est une limite théorique. Nous verrons plus tard, comment en pratique approcher cette limite. Les différents algorithmes de codage donnent plus ou moins satisfaction.

Exemple 1.3 (Exemple important : entropie d'une source binaire)**.** Soit une source émettant des symboles 0 avec la probabilité p et 1 avec la probabilité $q = 1 - p$. Alors l'entropie de cette source est

$$H_2(p) = -p \log_2(p) - (1-p) \log_2(1-p).$$

Lorsque, par exemple, les deux symboles sont équiprobables, on a $p = 1/2$ et alors

$$H_2(1/2) = -1/2\log_2(1/2) - 1/2\log_2(1/2) = 1.$$

Nous pouvons interpréter ce résultat comme suit : *lorsque les symboles d'une source binaire sont équiprobables, il faut un bit par symbole en moyenne.*

Il s'agit de la valeur maximale possible. Nous allons dans la suite utiliser cette fonction pour introduire ou illustrer les propriétés fondamentales de l'entropie.

Proposition 1.2 (Propriétés de l'entropie d'une source binaire)**.**
Soit $H_2(p),\ p \in [0,1]$ l'entropie d'une source binaire définie dans l'exemple ci-dessous. En tant que fonction de p elle a les propriétés suivantes :

1. *$H_2(p)$ est une fonction continue sur $]0,1[$ telle que*

$$\lim_{p\to 0+} H_2(p) = 0 = \lim_{p\to 1-} H_2(p)$$

2. *$H_2(p)$ est positive sur $H_2(p),\ p \in [0,1]$*
3. *$H_2(p)$ est symétrique par rapport à $p_0 = 0.5$ et atteint son maximum en p_0 t.q. $H_2(0.5) = 1$.*
4. *$H_2(p)$ est strictement concave :*

$$\forall (p_1, p_2) \in [0,1],\ p_1 \neq p_2, \quad \forall \lambda \in]0,1[$$

$$H_2(\lambda p_1 + (1-\lambda)p_2) > \lambda H_2(p_1) + (1-\lambda)H_2(p_2)$$

Toutes ces propriétés sont faciles à vérifier avec les méthodes simples de l'analyse des fonctions à une variable réelle. Elles sont résumées par la représentation graphique de $H_2(p)$ donnée sur la figure 1.1.

6.1 Interprétations de la fonction d'entropie

Considérons l'exemple d'une source S dont l'alphabet est composé de 5 symboles $\Omega = \{a, b, c, d, e\}$ et la distribution de probabilité est la suivante :

X	a	b	c	d	e
P(X)	0.3	0.2	0.2	0.15	0.15

L'entropie de cette source, calculée selon la définition est

$$H(X) = -\sum_{i=1}^{5} p_i \log_2(p_i) = -(0.3\log_2(0.3) - 2\cdot 0.2\log_2(0.2) - 2\cdot 0.15\log_2(0.15) \simeq 2.27.$$

Entropie comme mesure d'information. Nous avons déjà mentionné que l'entropie représente la quantité moyenne d'information apportée par l'observation des symboles de la source. On peut également l'interpréter comme la difficulté moyenne de prédire chaque symbole à la sortie.

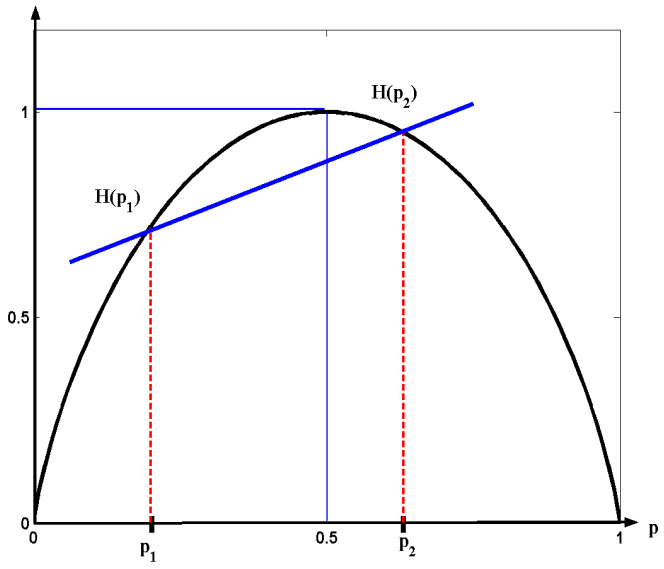

FIGURE 1.1 – Fonction d'entropie d'une source binaire

Entropie comme mesure de nombre de bits pour le codage. Imaginons que l'on essaye de deviner le symbole observé à la sortie de la source. Pour ce faire on peut poser des questions à un automate qui ne répond que par **"oui"** ou par **"non".**

Voici un schéma représentant le déroulement du jeu :

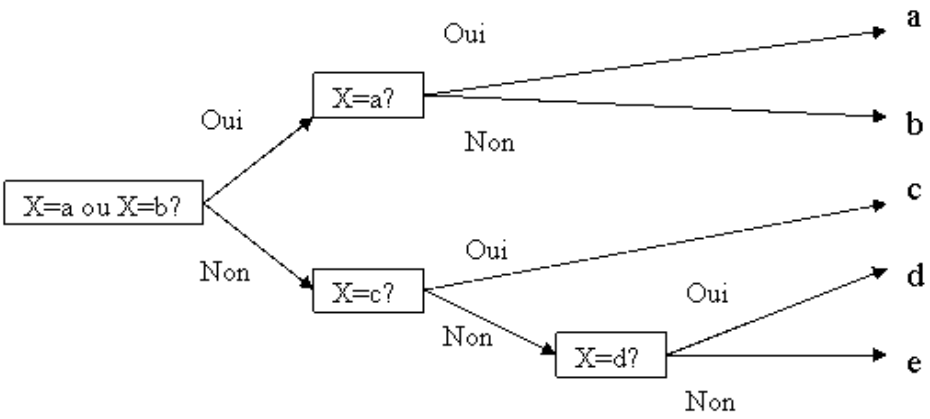

FIGURE 1.2 – Questions

Il est facile de voir que le nombre de questions nécessaires pour deviner le symbole est une

variable aléatoire Nb définie sur Ω de la façon suivante :

x	a	b	c	d	e
P(x)	0.3	0.2	0.2	0.15	0.15
Nb(x)	2	2	2	3	3

Alors le nombre moyen de questions nécessaires pour deviner le symbole est

$$\bar{Nb} = 2(0.3 + 0.2 + 0.2) + 3(0.15 + 0.15) = 2.3.$$

Si l'on utilise un bit pour coder la réponse à chaque question, cela nous permet de coder les symboles de cette source selon le schéma ci-dessus :

a	Oui-Oui	11
b	Oui-Non	10
c	Non-Oui	01
d	Non-Non-Oui	001
e	Non-Non-Non	000

Nous étudierons plus loin l'un des théorèmes fondamentaux de la théorie de l'information établissant que l'entropie d'une source est le plus petit nombre moyen de bits par symbole nécessaire pour coder les messages produits par cette source. Dans cet exemple, nous avons bien $\bar{Nb} = 2.3 > 2.27 = H(X)$.

6.2 Propriétés générales de la fonction d'entropie

Proposition 1.3 (Positivité)**.** *Soit S une source sans mémoire et stationnaire d'alphabet $\Omega = \{\omega_i\}_{i=1}^n$. Soit X la variable aléatoire associée de distribution de probabilité P donnée $P[X = \omega_i] = p_i, \quad i = 1, \ldots, n$. Notons $H(p_1, \ldots, p_n)$ sa fonction d'entropie. Alors*

$$H(p_1, p_2, \ldots, p_n) \geq 0.$$

En plus, l'égalité a lieu uniquement si l'une des probabilités p_i est égale à 1 et les autres sont nulles.

Preuve de la proposition 1.3

Sachant que $\forall i = 1, \ldots, n, \quad 0 \leq p_i \leq 1$ on a $-p_i \log_2(p_i) \geq 0$. Si l'un des symboles de l'alphabet est certain ($p_i = 1$) alors tous les autres sont forcément impossibles ($\forall k \neq i, \quad p_k = 0$).
On a alors : $p_i \log_2(p_i) = 0$ et $\forall k \neq i,\ p_k \log_2(p_k) = 0$ (par continuité) et donc $H = 0$.
C.Q.F.D

Lemme 1.1 (Inégalité de Gibbs)**.** *Soient $P = (p_i)_{i=1}^n$ et $Q = (q_i)_{i=1}^n$ deux distributions de probabilités sur le même univers fini $\Omega = \{\omega_i\}_{i=1}^n$. Alors*

$$\sum_{i=1}^{n} p_i \log_2 \left(\frac{p_i}{q_i} \right) \leq 0 \tag{1.1}$$

et l'égalité a lieu ***si et seulement si*** $\forall i = 1, \ldots, n, p_i = q_i$.

Théorème 1.1 (Maximum de la fonction d'entropie)**.**

$$H(p_1, p_2, \ldots, p_n) \leq log(n) \tag{1.2}$$

et l'égalité a lieu si et seulement si $\forall i = 1, \ldots, n, \quad p_i = \frac{1}{n}$.

Preuve du théorème 1.1

Soient $P = (p_i)_{i=1}^n$ une distribution de probabilité sur l'univers fini

$$\Omega = \{\omega_i\}_{i=1}^n$$

associé à une source et $Q = \left(\frac{1}{n}\right)_{i=1}^n$ la distribution uniforme sur le même univers. Appliquons l'inégalité de Gibbs à P et Q :

$$\sum_{i=1}^{n} p_i \log_2 \left(\frac{p_i}{q_i}\right) \leq 0 \;\Leftrightarrow\; \sum_{i=1}^{n} p_i \log_2 (np_i) \leq 0$$

d'où

$$0 \geq \sum_{i=1}^{n} p_i(\log_2(n)+\log_2(p_i)) = \log_2(n)\sum_{i=1}^{n} p_i+\sum_{i=1}^{n} p_i \log_2(p_i) = \log_2(n)-H(p_1, \ldots, p_n)$$

D'après le lemme précédent, l'égalité a lieu si et seulement si $\forall i = 1, \ldots, n, p_i = q_i = \frac{1}{n}$.

C.Q.F.D

Remarque 1.3. Nous pouvons interpréter ce résultat en disant que l'incertitude sur le symbole observé à la sortie d'une source est maximale lorsque tous les symboles sont équiprobables. Nous retrouvons ici le cas particulier de l'entropie d'une source binaire $H_2(p)$ étudiée dans la proposition 1.2.

Proposition 1.4 (Concavité). *Soient $P = (p_i)_{i=1}^n$ et $Q = (q_i)_{i=1}^n$ deux distributions de probabilités sur le même univers fini $\Omega = \{\omega_i\}_{i=1}^n$ associé à une source S. Alors Alors, $\forall \lambda \in [0,1]$*

$$H(\lambda P + (1-\lambda)Q) \geq \lambda H(P) + (1-\lambda)H(Q)$$

ou encore

$$H(\lambda p_1 + (1-\lambda)q_1, \ldots, \lambda p_n + (1-\lambda)q_n \geq \lambda H(p_1, \ldots, p_n) + (1-\lambda)H(q_1, \ldots, q_n).$$

De plus, l'égalité a lieu si et seulement si

$$\lambda \in \{0,1\} \text{ ou } P = Q$$

Remarque 1.4. Nous pouvons interpréter cette propriété comme suit. L'entropie d'une moyenne de deux sources est supérieure à la moyenne de leurs entropies.

Exemple 1.4. Considérons trois sources binaires S_1, S_2 et S_3 sur l'alphabet $\Omega = \{a, b\}$. Supposons que les distributions de probabilités de S_1 et S_2 sont les suivantes

$$P_1(a) = p_1 = \frac{1}{3},\ P_1(b) = (1-p_1) = \frac{2}{3}$$

$$P_2(a) = p_2 = \frac{1}{2},\ P_2(b) = (1-p_2) = \frac{1}{2}$$

Supposons que la distribution de S_3 soit la moyenne des deux premières

$$p_3 = \frac{p_1 + p_2}{2} = \frac{5}{12}$$

Calculons les entropies des trois sources

$$H(S_1) = H_2(p_1) = -p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1) = -\frac{2}{3} + \log_2(3)$$

$$H(S_2) = H_2(p_2) = -p_2 \log_2(p_2) - (1-p_2)\log_2(1-p_2) = 1$$

$$H(S_3) = H_2(p_3) = -p_3 \log_2(p_3) - (1-p_3)\log_2(1-p_3) \simeq 0.979868$$

On constate que l'entropie de la troisième source est supérieure à la moyenne des entropies des deux premières :

$$H(S_3) = 0.979868 > \frac{H(S_1) + H(S_2)}{2} = \frac{1 - \frac{2}{3} + \log_2(3)}{2} \simeq 0.9591479$$

Chapitre 2

Premier domaine d'application : cryptographie

Ce chapitre n'a pas pour ambition de faire un exposé complet sur la cryptographie. Nous allons ici montrer en quoi les principaux concepts de la théorie de l'information que nous venons d'introduire sont utiles dans ce domaine.

En 1949 C. Shannon a publié un article intitulé "Communication Theory of Secrecy Systems" dans la revue Bell System Technical Journal. Dans cet ouvrage C. Shannon a montré comment, à l'aide de la notion d'entropie, mesurer la sécurité d'un système de communications crypté. Il a introduit la notion de cryptosystème parfaitement sûr et a mis en évidence les faiblesses d'un tel système dues aux difficultés de sa mise en pratique. Nous suivrons son raisonnement pour arriver à montrer l'importance du concept d'entropie dans ce domaine.

Pour cela nous commencerons par une brève présentation des notions élémentaires de la cryptographie. Ensuite, nous allons exposer la théorie de C. Shannon sur la sécurité d'un cryptosystème.

1 Quelques définitions

La **cryptographie** est une science qui étudie les méthodes permettant de transmettre des messages de façon confidentielle.

On va appeler **message clair** un ensemble de données (texte, image,...) que l'on souhaite transmettre.

Le **chiffrement** est un procédé permettant de transformer le message clair de telle sorte qu'il soit incompréhensible par qui que ce soit d'autre que l'auteur du message et le destinataire.

Le **chiffré** est le résultat du chiffrement.

Le **déchiffrement** est le procédé permettant de retrouver le message clair à partir du chiffré.

La **clé** de chiffrement est un paramètre qui est utilisé dans le procédé de chiffrement ou déchiffrement.

Un **cryptosystème** est un ensemble de clés $\mathcal{K}$, de messages clairs possibles , $\mathcal{M}$, de messages chiffrés possibles $\mathcal{C}$ associée à un algorithme de chiffrage, représenté par une fonction $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, et un procédé de déchiffrage , représenté par une fonction $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$. On suppose que pour tout $m \in \mathcal{M}$ il existe une paire de clés de chiffrement et de déchiffrement telles que la relation

$$D(k_D, E(k_E, m)) = m$$

est assurée. On note $(\mathcal{K}, \mathcal{M}, \mathcal{C}, E, D)$ un cryptosystème.

Remarque 2.1 (Principe de Kerckhoffs)**.** De nos jours, la plus grande partie de méthodes de cryptographie reposent sur ce principe, énoncé par Auguste Kerckhoffs à la fin XIXe siècle (publié dans un article au Journal des sciences militaires, vol. IX, pp. 5-38, Janvier 1883, pp. 161-191, Février 1883. Source : Wikipédia).

La sécurité d'un cryptosystème ne doit pas reposer sur la non divulgation de la fonction de cryptage mais uniquement sur la non divulgation de la clé.

Un autre énoncé de ce principe, connu sous le nom de maxime de Shannon a été proposé par Claude Shannon, au milieu du XXe sciècle : **L'adversaire connaît le système.**

Deux modèles principaux de chiffrement existent actuellement :

Chiffrement symétrique On appelle aussi ce modèle "chiffrement à clé secrète". C'est l'approche classique. La même clé est utilisée pour chiffrer et déchiffrer un message. Cette clé doit alors rester secrète. Dans les systèmes de communication à grande échelle cela représente la principale faiblesse de ces chiffrements. En effet la clé doit être communiquée au destinataire et dans de nombreux cas elle est envoyée par le même canal de transmission que le message lui même. Si elle est interceptée à ce moment là, toute la communication est découverte !

Chiffrement assymétrique. On l'appelle aussi chiffrement à clé publique. La clé utilisée pour le chiffrement est publique et elle est différente de la clé de déchiffrement, qui est secrète. Ainsi le destinataire du message choisit la clé de déchiffrement et la clé de chiffrement associée. Il peut envoyer à l'émetteur la clé de chiffrement (publique). Et la clé de déchiffrement, secrète, ne circule pas dans les canaux de transmission.

1.1 Quelques exemples élémentaires

1.1.1 Chiffrement de César (50 av. J.C.)

Le chiffre de César utilisait le principe de décalage cyclique de l'alphabet. Soit en effet l'alphabet latin A de 26 lettres. Un message $m = m_1 m_2 \dots m_n$ est chiffré en remplaçant chaque lettre m_i par la lettre $\sigma(m_i)$ obtenue par décalage cyclique de trois positions dans l'alphabet. Par exemple, la lettre "a" sera remplacée par la lettre "d", la lettre "z" par "c". On peut utiliser ce principe avec un décalage de k caractères, $k = 1, \dots, 26$. Le paramètre de décalage est alors la clé secrète. Ce système est facile à casser car l'espace des clés est très réduit : ici il n'y a que 26 clés possibles. Il suffit alors de tester toutes les 26 possibilités jusqu'à trouver un texte intelligible.

Il est possible de rendre le chiffrage par substitution plus complexe. Au lieu d'utiliser les décalages cycliques, on peut utiliser une permutation quelconque $\sigma : A \to A$. Alors la fonction de chiffrement se présente de façon suivante. Soit un message $m = m_1 \dots m_n$.

$$E(m) = E(\sigma, m) = \sigma(m_1)\sigma(m_2) \dots \sigma(m_n)$$

La clé secrète est alors la permutation σ et l'ensemble $\mathcal{K}$ de toutes les clé est celui de toutes les permutations possibles sur l'alphabet de 26 caractères. On a donc $Card(\mathcal{K}) = 26! \simeq 4 \cdot 10^26$. Il est donc raisonnable de supposer qu'une attaque par force brute (qui consiste à tester toutes les clés possibles, comme ci-dessus) ne soit pas efficace.

Il est néanmoins possible de casser un tel code par analyse statistique. En effet, si l'on connaît la langue du message clair, on connaît la table de fréquences des caractères. Par exemple, en français, la lettre "e" est la plus fréquente. On peut également utiliser les tables de fréquences d'ordres supérieurs qui établissent les probabilités d'occurrence de couples de caractères. En français par exemple, le caractère "u" a 95% de chances d'être trouvé après "q". Alors, en analysant le chiffré seul et en calculant les fréquences d'occurrence des différents caractères dans le chiffré on peut retrouver la permutation et donc le texte clair. C'est un exemple d'attaque à chiffré seul.

1.1.2 Substitution polyalphabétique

Une variante de méthode de substitution consiste à utiliser un nombre fini de permutations au lieu d'une seule. On découpe alors le texte clair en blocs de p caractères et on choisit p permutations d'alphabet : $\sigma_1, \dots, \sigma_p$. Alors pour chaque bloque de p caractères de texte on applique la transformation

$$E(m_1 \dots m_p) = \sigma_1(m_1) \dots \sigma_p(m_p)$$

Par exemple, le chiffrement de la machine Enigma utilisée pendant la deuxième guerre mondiale était un chiffrement de ce type avec des décalages cycliques pour chaque permutation. La clé était alors une suite de nombres dont la longueur indiquait la longueur des blocs et chaque nombre représentait le décalage à appliquer au caractère correspondant du bloc.

Ce chiffrement, bien que plus complexe que les précédents peut aussi être cassé par les méthodes statistiques basées sur l'étude des fréquences d'apparition des caractères.

1.1.3 Chiffrement de Vernam

Cette méthode a été proposée en 1917. Elle a été longtemps utilisée pour le chiffrement des messages télégraphiques. Les messages clairs sont des suites de bits de longueur n. L'espace des messages est alors $\mathcal{M} = 0, 1^n$. Les clés sont les suites binaires de même longueur que les messages : $\mathcal{K} = \mathcal{M} = 0, 1^n$. La fonction de chiffrage consiste à ajouter bit à bit module 2 la clé au message clair. Pour tout $m = m_1 \dots m_n$ et pour tout $k = k_1 \dots k_n$

$$c = E(k, m) = k \oplus m = m_1 \oplus k_1 \dots m_n \oplus k_n$$

Nous allons expliquer dans la section suivante pourquoi ce chiffrement est parfaitement sûr au sens de Shannon. Remarquons pour l'instant ses faiblesses. Ce chiffrement est vulnérable devant

une attaque à clair connu : si quelqu'un découvre ne serait ce qu'un bloc de message clair m il peut immédiatement obtenir la clé en la calculant :

$$k = m \oplus c$$

Il est alors indispensable de changer de clé à chaque bloc. C'est pour cela qu'on appelle cette méthode "masque jetable". Si l'on prend en compte le fait que les clés sont de même longueur que les blocs et qu'il faut les transmettre par un canal sécurisé avant de transmettre le chiffré il devient évident que la mise en pratique de cette méthode est difficile. Ce code a tout de même été utilisé pour le téléphone rouge. La clé alors été envoyée par porteur car il était préférable d'envoyer une nouvelle clé si elle a été interceptée que de courir le risque que le message secret soit intercepté et décodé.

2 La théorie de C. Shannon sur la sécurité entropique.

Nous allons maintenant nous intéresser à la signification que prend le concept d'entropie dans le contexte de cryptographie.

Rappelons que l'entropie d'une variable aléatoire mesure l'incertitude associée au résultat d'une expérience que la variable représente. Nous allons donc donner une interprétation probabiliste au problème de chiffrement en nous plaçant à la place de cryptanalyste, celui qui cherche par analyse à retrouver le message clair à partir du chiffré. De son point de vue, le choix d'un message clair et celui d'une clé sont des résultats d'un tirage aléatoire. Le message chiffré est alors aussi aléatoire. Soient donc les variables aléatoires M, C et K représentant respectivement le choix d'un message dans l'ensemble $\mathcal{M}$, le choix d'un chiffré dans l'ensemble $\mathcal{C}$ et celui d'une clé dans l'ensemble $\mathcal{K}$. C. Shannon définit un cryptosystème parfaitement sûr comme étant le système dans lequel la connaissance du chiffré n'apporte pas d'information sur le message clair.

Définition 2.1 (Cryptosystème parfaitement sûr)**.** Soit un cryptosystème $(\mathcal{K}, \mathcal{M}, \mathcal{C}, E, D)$. Soient M et C les variables aléatoires représentant le choix d'un message claire et d'un chiffré. Le système est dit parfaitement sûr ssi

$$H(M|C) = H(M)$$

Proposition 2.1 (La sécurité parfaite du chiffre de Vernam)**.** *Supposons que l'espace des clés est le même que l'espace des messages clairs :*$\mathcal{M} = \mathcal{K}$ *et que toutes les clés sont équiprobables. Alors le chiffre de Vernam est parfaitement sûr.*

Preuve de la proposition 2.1

Nous allons supposer sans perte de généralité que l'alphabet utilisé est binaire et que les messages et les clés sont des séquences de bits de longueur connue n. Alors l'espace des messages clairs et des clés est

$$\mathcal{M} = \mathcal{K} = \{0,1\}^n$$

Comme toutes les clés sont supposées équiprobables, on a

$$\forall k \in \mathcal{K},\ P[K = k] = \frac{1}{2^n}$$

Pour établir l'égalité des entropies il suffit de montrer que

$$\forall m \in \mathcal{M},\ \forall c \in \mathcal{C},\ P(M = m|C = c) = P(M = m)$$

Par définition de probabilité conditionnelle on a

$$P(M = m|C = c) = \frac{P(M = m, C = c)}{P(C = c)}$$

On sais que dans le chiffrement de Vernam les trois variables K, M et C sont liées : $C = K \oplus M$. Alors

$$C = c \wedge M = m \ \Leftrightarrow\ K = c - m \wedge M = m$$

Donc on a

$$P(M = m, C = c) = P(M = m, K = c-m) = P(M = m)P(K = c-m) = \frac{P(M = m)}{2^n}$$

car les variables aléatoires K et M sont indépendantes.
On calcule maintenant la probabilité

$$P(C = c) = \sum_{x \in \mathcal{M}} P(C = c, M = x) = \sum_{x \in \mathcal{M}} \frac{P(M = x)}{2^n} = \frac{1}{2^n}$$

C.Q.F.D

Le théorème suivant permet de montrer qu'un système parfaitement sûr doit avoir des clés au moins aussi longues que les messages clairs.

Théorème 2.1. *Dans un système cryptographique parfaitement sûr on a*

$$H(K) \geq H(M)$$

En particulier, si tous les messages et toutes les clés sont équiprobables, les clés sont de longueur au moins égale à celle des messages.

Il n'est donc pas possible de trouver un chiffrement parfaitement sûr ayant des clés moins longues que celles du chiffre de Vernam.

Dans ses travaux C. Shannon s'est aussi intéressé aux systèmes à clé courte, qui ne sont pas parfaits. La question que l'on peut alors poser est la suivante. Etant donné un système cryptographique dans lequel toutes les clés et tous les messages clairs sont équiprobables, de combien de chiffrés faut il disposer pour être capable de retrouver de façon unique la clé ? Pour répondre à la question nous allons introduire quelques notions supplémentaires.

Définition 2.2. Soit un langage composé de mots de longueur donnée N sur un alphabet donné A. On associe la variable aléatoire M au choix au hasard d'un mot du langage. On appelle taux du langage ou taux d'entropie, la quantité

$$r = \lim_{N \to \infty} \frac{H(M)}{N}$$

Si le nombre de caractères de l'alphabet est L on appelle taux maximal du langage la quantité

$$R = log_2 L$$

Enfin on appelle redondance du langage la différence

$$D = R - r$$

Dans cette définition, le taux du langage représente la quantité moyenne d'information par caractère de message, le taux maximal correspond à l'entropie maximale d'un caractère, c'est à dire au cas où tous les messages sont équiprobables. Enfin, la redondance représente la différence entre le nombre de bits maximal et ce qui est tout juste nécessaire pour coder un caractère. Par exemple, le taux de l'anglais a été évalué par méthodes statistiques près de 1.5.

Nous allons maintenant introduire la mesure permettant de quantifier la quantité d'information dont il est faut disposer pour pouvoir déterminer la clé d'un chiffrement.

Définition 2.3 (Distance d'unicité)**.** Soit un cryptosystème $(\mathcal{K}, \mathcal{M}, \mathcal{C}, E, D)$. On appelle la distance d'unicité le plus petit nombre de messages chiffrés dont il est nécessaire disposer pour que l'incertitude résiduelle sur la clé sachant ces chiffrés soit nulle :

$$n_0 : \ H(K|C_1, \ldots, C_{n_0}) = 0$$

On peut montrer la proposition suivante

Proposition 2.2. *La distance d'unicité d d'un cryptosystème est égale à*

$$d = \frac{H(K)}{D} = \frac{H(K)}{R - r}$$

où D est la redondance du langage.

Exemple 2.1. On a établi par analyse statistique que le taux d'entropie du français est d'environ $r \simeq 3.97$ et que le taux maximal est $R = 4.67$. Ainsi la redondance est $D = 0.7$. On peut donc calculer la distance d'unicité pour un chiffrement par substitution. L'espace de clés est alors l'ensemble de toutes les permutations possibles de l'alphabet latin et si on considère que les clés sont équiprobables on a

$$d = \frac{H(K)}{D} = \frac{log_2(26!)}{0.7} \simeq 126$$

Il faut donc en moyenne 126 caractères de texte chiffré pour être en mesure de déterminer la clé de façon unique

Remarque 2.2. Il s'agit bien d'une condition nécessaire à la possibilité de découvrir la clé. Il faut interpréter ce résultat dans le sens suivant : si l'on ne dispose pas de longueur suffisante de message chiffré donnée par la distance d'unicité, il est impossible de déterminer la clé avec certitude.

Ce résultat ne permet surtout pas de se prononcer sur la puissance de calcul nécessaire pour découvrir la clé ni sur les moyens d'y parvenir.

Remarque 2.3. Il est important d'observer que la distance d'unicité est inversement proportionnelle à la redondance des messages clairs. Plus l'ensemble des messages clairs est redondant moins il faut d'informations pour déterminer la clé. Ainsi la sécurité du système dépend aussi de la qualité intrinsèque des messages.

Ainsi, pour améliorer la distance d'unicité, il est préférable de traiter en amont les messages de façon à réduire la redondance. Par exemple, une compression sans pertes par codage de Huffman ou autre peut être utilisée avant le chiffrement. Nous allons aborder ces techniques dans les chapitres suivants.

Chapitre 3

Codage de source.

1 Codage

D'une manière générale le codage peut être vu comme une transformation de symboles d'un alphabet donné $\Omega_1 = \{s_1, \ldots, s_n\}$ en suites de symboles d'un autre alphabet $\Omega_2 = \{c_1, \ldots, c_d\}$.

Dans un schéma de communication, la source et le canal de transmission n'utilisent pas forcément le même alphabet. Par exemple, la source peut être un texte anglais, utilisant les 26 lettres de l'alphabet latin, et le canal peut être tout support numérique, utilisant l'alphabet binaire. Il se pose donc le problème de **codage** comme "traduction" entre l'alphabet de la source et celui du canal. Si le récepteur utilise le même alphabet que la source il est également de **décoder** le message avant de pouvoir le lire, c'est-à-dire, appliquer la transformation inverse. Toute transformation candidate doit vérifier un certain nombre de critères que nous allons détailler plus loin. Étant donné qu'il peut exister une grande quantité de codages possibles il se posera la question de savoir lequel est meilleur que tous les autres. Cela dépendra de critère d'optimalité que l'on fixera. Nous allons dans cette perspective considérer deux types d'applications de codage :

1. **Codage de source ou encore codage sans bruit.** Nous allons supposer que la communication se fait via un canal sans bruit. Cela signifie que tout message est transmis de façon exacte. Sous cette hypothèse le meilleur code sera celui qui permettra la transmission la plus rapide possible. **Le premier théorème de Shannon** donne la solution à ce problème. *A noter* que ce cours sera dans la suite consacré essentiellement à l'étude des méthodes de codage de source.
2. **Codage de canal ou encore codage en présence de bruit.** En supposant qu'il existe des perturbations pouvant engendré des erreurs à la réception, on cherchera une méthode de codage permettant une transmission aussi rapide que possible tout en minimisant la probabilité des erreurs. **Le second théorème de Shannon** donne la solution à ce problème.

2 Premier théorème fondamental

2.1 Codage de source

Nous allons dans un premier temps considérer un modèle de communication avec un canal discret sans bruit. Soient $\Omega_S = \{s_1, \ldots, s_n\}$ l'alphabet de la source et $\Omega_C = \{c_1, \ldots, c_d\}$ l'alphabet du canal. Supposons que la distribution de probabilité de l'alphabet de la source est connue :

$$P_S : \Omega_S \to [0,1], \quad s_i \mapsto p_S(s_i)$$

On peut alors en déduire l'entropie de la source

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Nous utiliserons dans la suite la terminologie suivante :

Lettre, symbole ou caractère Tout élément d'un alphabet donné ;

Message ou mot Une séquence fini m de caractères d'un alphabet donné ;

Longueur de mot le nombre $l(m)$ de caractères d'un mot m ;

Le codage consiste à faire correspondre à chaque symbole s_i de la source une séquence $m(s_i)$ de symboles de l'alphabet du canal, appelée **mot-code**. Une telle association peut donc être représentée par un ensemble $\{m_1, m_2, \ldots, m_n\}$ de n mots codes correspondants chacun à un symbole de l'alphabet de la source : $\forall\ i = 1, \ldots, n,\ m_i = m(s_i)$. Si l'on note l_i les longueurs des mots m_i du code. Soit L la variable aléatoire dont la valeur est la longueur du mot-code associé à un symbole émis par la source S. Alors cette variable aléatoire prend les valeurs l_i avec les probabilités $p_i = p_S(s_i), \quad i = 1, \ldots, n$. On définit alors la longueur moyenne du code par

$$\bar{L} = E[L] = \sum_{i=1}^{n} p_i l_i$$

Cette quantité est importante pour l'analyse des caractéristiques d'un code donné. Elle représente le nombre moyen de caractères de l'alphabet Ω_C pour coder un symbole émis par la source S.

Supposons que le temps de transmission est le même pour tous les caractères de l'alphabet du canal. Alors, le temps moyen de transmission d'un symbole de l'alphabet de la source est proportionnel à la longueur moyenne des mots du code, $\bar{L}$. C'est pour cette raison que nous allons dans la suite étudier en détail ce paramètre.

Un code doit vérifier un certain nombre de propriétés garantissant la possibilité de reconstituer tout message codé à la réception. Ces propriétés sont :

Régularité Un code $\{m_1, m_2, \ldots, m_n\}$ est dit régulier si tous les mots qui le composent sont distincts : $m_i \neq m_k, \quad \forall i \neq k$. Cette condition garantit au moins que tout message d'un seul caractère de l'alphabet de la source peut être décodé. Un code qui n'est pas régulier est dit singulier ou irréversible.

Déchiffrabilité Un code régulier est dit déchiffrable (ou encore à décodage unique) si pour toute suite de mots de code $m^1 m^2 \ldots m^k$ il est possible de distinguer sans ambiguïté tous les mots et donc identifier les symboles $s^j,\ j = 1, \ldots, k$ composant le message.

Voici un exemple de plusieurs codes possibles pour un même alphabet.

Exemple 3.1. Soit $\Omega_S = \{a, b, c, d\}$ de distribution de probabilité $P_S = \{0.4, 0.3, 0.2, 0.1\}$. L'entropie de cette source est $H(S) \simeq 1.85$. Supposons que le canal est binaire. Le tableau ci-dessous propose quelques codes et leurs longueurs moyennes de mots :

S	Proba	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6
a	0.4	1	00	0	0	0	0
b	0.3	0	01	10	01	10	11
c	0.2	1	10	01	011	110	100
d	0.1	0	11	010	0111	1110	101
	Long. Moy.	1	2	1.7	2	2	1.9

Le code 1 n'est pas régulier. En effet, le caractère 0 correspond à deux caractères différents dans l'alphabet initial. Tous les autres codes du tableau sont régulier. Le code 2 est un code **de longueur fixe** : tous les mots du code sont de même longueur. Les codes 3-6 sont de longueur variable.

Le code 3 est régulier mais pas déchiffrable. En effet, la déchiffrabilité signifie que toute séquence de mots du code correspond à au plus un message. Dans le cas du code 3, la séquence 010 correspond à la fois à trois messages différents : "d", "ca" et "ab". Elle ne peut donc pas être décodée correctement. Nous allons maintenant analyser le problème de déchiffrabilité en détail.

2.1.1 Le problème de décodage unique

Il existe plusieurs façons de garantie qu'un code soit déchiffrable :

Codes de longueur fixe Un code régulier de longueur fixe peut toujours être décodé sans ambiguïté car il suffit pour cela de découper la séquence en mots de longueur connue. Cette solution présente néanmoins un désavantage. On peut l'observer dans le tableau ci-dessus. Le code 1, de longueur fixe a la longueur moyenne de code égale à 2 tandis qu'il existe dans la même table des codes avec une longueur moyenne inférieure.

Utilisation d'un séparateur Il est possible de consacrer un symbole de l'alphabet du canal comme séparateur de mots du code. Par exemple, pour un canal binaire, on peut coder le i-ème symbole de la source s_i à l'aide de i caractères "1" et utiliser "0" comme séparateur. Dans le cas de l'exemple 3.1 cela donnerait le code suivant

S	Proba	Code 1
a	0.4	10
b	0.3	110
c	0.2	1110
d	0.1	11110
	Long. Moy.	3

et la séquence "abc" donnerait "101101110"

On constate que la longueur moyenne qui tient compte du séparateur est plus élevée que tous les autres codes.

Codes sans préfixe On dit qu'un mot W est **un préfixe** d'un autre mot V s'il existe un mot U tel que $V = WU$. Autrement dit, le mot V commence par le mot W. Dans ce cas le mot U s'appelle *suffixe*.

On dit qu'un code donné est **sans préfixe** ou **instantané** si aucun mot du code n'est un préfixe d'un autre.

Dans notre exemple 3.1 le code 6 est sans préfixe. Un tel code est toujours déchiffrable. En effet, pour décoder une séquence quelconque de mots du code il suffit lire la séquence caractère par caractère de gauche à droite. Dès qu'un mot du code m est formé on sait qu'il n'est pas un début d'un autre mot. On peut donc séparer le mot et recommencer la lecture. Cela donne une procédure de décodage "pas à pas". Prenons une séquence $W = 011010011101$ du code 6. Le décodage de la séquence se passe de la façon suivante :

Pas 1 Le premier mot du code formé en lisant de gauche à droite est $m^1 =$ "0". Donc le premier symbole est $s^1 = a$. On sépare le mot m^1 de la séquence. On obtient la nouvelle séquence $W_1 = 11010011101$.

Pas 2 $m^2 =$ "11" $\Rightarrow$ $s^2 = b$ et $W_2 = 010011101$.

Pas 3 $m^3 =$ "0" $\Rightarrow$ $s^3 = a$ et $W_3 = 10011101$.

Pas 4 $m^4 =$ "100" $\Rightarrow$ $s^4 = c$ et $W_2 = 11101$.

Pas 5 $m^5 =$ "11" $\Rightarrow$ $s^5 = b$ et $W_2 = 101$.

Pas 6 $m^6 =$ "101" $\Rightarrow$ $s^6 = d$ et $W_6 = \emptyset$.

On obtient en symboles de l'alphabet de la source : *abacbd*.

On remarque que dans la famille de codes à longueur variable et sans séparateur les codes sans préfixe ou instantanés représentent un intérêt. Il est évident que tout code instantané est déchiffrable.

La réciproque n'est pas vraie. Soit en effet une source binaire d'alphabet $\Omega_S = \{a, b\}$ et un canal binaire d'alphabet $\Omega_C = \{0, 1\}$. Le code suivant $m_1 = m(a) = 0$, $m_2 = m(b) = 01$ n'est pas instantané car m_1 est un préfixe de m_2. Il est tout de même déchiffrable. Pour décoder une séquence de mots d ece code il suffit de repérer d'abord les positions de "1". Chaque "1" est obligatoirement précédé de "0" et donne le caractère "b". les autres "0" correspondent à "a". C'est comme cela qu'on trouve que la séquence "000101000100" correspond à "aabbaabaa".

On voit alors la signification du terme "instantané" et l'intérêt principal de ces codes. Un code sans préfixe peut être décodé pas à pas, par séparation successive des mots. Cela peut être fait au fur et à mesure de la réception du message. Tandis qu'un code déchiffrable mais non instantané nécessite un traitement plus long et spécifique au code pour être déchiffré.

Nous allons nous intéresser maintenant au problème d'existence de codes instantanés. Ce problème se formule de la façon suivante

> Soient l'alphabet de la source $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et et l'alphabet du canal $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d. Étant donnés n nombres entiers positifs $(l_1, l_2, \ldots, l_n) \in \mathbb{Z}_+^*$ existe-t-il un code régulier instantané de n mots $\{m_1, \ldots, m_n\}$ tel que chaque nombre l_i soit la longueur du mot de code m_i ?

Le théorème suivant donne la condition nécessaire et suffisante d'existence de tels codes.

Théorème 3.1 (Inégalité de Kraft)**.** *Un code instantané de longueurs de mots données $l_1, \ldots, l_n$ existe si et seulement si*

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du canal.

Historiquement, l'inégalité de Kraft a d'abord été démontrée par McMillan, comme condition nécessaire et suffisante d'existence de codes déchiffrables de longueurs de mots données. Voici le théorème qui est une extension du théorème précédent sur la classe entière de codes déchiffrables :

Théorème 3.2 (Condition de McMillan)**.** *Un code déchiffrable de longueurs de mots données $l_1, \ldots, l_n$ existe si et seulement si*

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du canal.

Remarque 3.1. Ces deux théorèmes montrent qu'il existe un code déchiffrable de longueurs de mots données $l_1, \ldots, l_n$ si et seulement si il existe un code instantané de mêmes longueurs de mots.

Nous avons maintenant pourvoir poser le problème de codage de source ou encore de codage sans bruit :

Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Existe-t-il un code qui minimise la longueur moyenne de mots $\bar{L}$?

2.2 Le premier théorème de Shannon

Nous allons approcher la solution du problème de codage de source en trois étapes. D'abord, nous allons énoncer la borne inférieure pour la longueur moyenne de mots de code. Ensuite, nous allons proposer une borne supérieure. Et enfin, énoncer le premier théorème fondamental de la théorie de l'information qui montre qu'il est possible d'approcher la borne inférieure avec autant de précision que l'on souhaite.

2.2.1 Borne inférieure de longueur moyenne de code

Théorème 3.3 (Borne inférieure)**.** *Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Soit un code déchiffrable $\{m_1, \ldots, m_n\}$ de longueurs de mots $\{l_1, \ldots, l_n\}$.*
Alors la longueur moyenne de mots de code vérifie :

$$\bar{L} = \sum_{i=1}^{n} p(s_i) l_i \geq \frac{H(S)}{\log_2(d)}$$

L'égalité n'est possible que si $\forall i = 1, \ldots, n, \quad p_i = d^{-l_i}$.

Preuve du théorème 3.3

Puisque le code que nous avons est déchiffrable il vérifie l'inégalité de Kraft :

$$0 < Q = \sum_{i=1}^{n} d^{-l_i} \leq 1$$

On peut alors définir les nombres

$$q_i = \frac{d^{-l_i}}{Q}, \quad i = 1, \ldots, n$$

On a alors : $\sum_{i=1}^{n} q_i = 1$ et donc on peut appliquer l'inégalité de Gibbs (voir lemme 1.1)

$$\sum_{i=1}^{n} p_i \log_2 \left(\frac{p_i}{q_i} \right) \leq 0$$

et l'égalité a lieu **si et seulement si** $\forall i = 1, \ldots, n, p_i = q_i$.

On en déduit alors :

$$
\begin{aligned}
-\sum_{i=1}^{n} p_i \log_2 p_i &\leq -\sum_{i=1}^{n} p_i \log_2 q_i && (3.1)\\
&= -\sum_{i=1}^{n} p_i \log_2 \frac{d^{-l_i}}{Q} = && (3.2)\\
&= -\sum_{i=1}^{n} p_i \log_2 d^{-l_i} + \sum_{i=1}^{n} p_i \log_2 Q && (3.3)\\
&= \sum_{i=1}^{n} p_i l_i \log_2 d + \log_2 Q \sum_{i=1}^{n} p_i && (3.4)
\end{aligned}
$$

En remarquant que $\sum_{i=1}^{n} p_i l_i = \bar{L}$ et que $\sum_{i=1}^{n} p_i = 1$ on a

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \leq \bar{L} + \log_2 Q$$

Enfin, d'après l'inégalité de Kraft que nous avons mentionné au début, $Q \leq 1$. Donc $\log_2 Q \leq 0$ d'où l'inégalité que nous devions démontrer.
C.Q.F.D

Remarque 3.2. On peut noter le fait que la quantité $\frac{H(S)}{\log_2(d)}$ représente l'entropie de la source calculée par rapport à la base d :

$$\frac{H(S)}{\log_2(d)} = -\sum_{i=1}^{n} p_i \frac{\log_2(p_i)}{\log_2(d)} = -\sum_{i=1}^{n} p_i \log_d(p_i) = H_d(S)$$

Un code dont la longueur moyenne de mots atteint la borne inférieure s'appelle **absolument optimal**. Un exemple de code absolument optimal est donné par la tableau suivant

S	Proba	Code
a	0.5	0
b	0.25	10
c	0.125	110
d	0.125	111

et on a $H(S) = \bar{L} = \frac{7}{4}$.

2.2.2 Borne supérieure de longueur moyenne de code

Cependant, un code absolument optimal n'est pas toujours réalisable. En effet, pour atteindre la borne inférieure, les longueurs de mots doivent vérifier $p_i = d^{-l_i}$ et donc $l_i = \frac{logp_i}{logd}$. Or ces

nombres ne sont pas forcément entiers. Dans ces cas la meilleure solution consiste à choisir les longueurs de mots de telle sorte que

$$\forall\ i = 1, \ldots, n, \quad -\frac{\log_2 p_i}{\log_2(d)} \leq l_i < -\frac{\log_2 p_i}{\log_2 d} + 1$$

Théorème 3.4 (Borne supérieure)**.** *Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire.*
Alors il existe un code déchiffrable dont la longueur moyenne de mots de code vérifie :

$$\frac{H(S)}{\log_2(d)} \leq \bar{L} = \sum_{i=1}^{n} p(s_i) l_i < \frac{H(S)}{\log_2(d)} + 1$$

Preuve du théorème 3.4

Choisissons les longueurs de mots comme précisé ci-dessus :

$$\forall\ i = 1, \ldots, n, \quad -\frac{\log_2 p_i}{\log_2 d} \leq l_i < -\frac{\log_2 p_i}{\log_2 d} + 1$$

Tout d'abord, montrons que l'inégalité de Kraft est vérifiée. En effet, de l'inégalité ci-dessus on déduit que

$$\forall\ i = 1, \ldots, n, \quad \log_2 p_i \geq -l_i \log_2 d \quad \Rightarrow \quad p_i \geq d^{-l_i}$$

Alors pour la somme on a

$$\sum_{i=1}^{n} d^{-l_i} \leq \sum_{i=1}^{n} p_i = 1$$

Alors il existe un code instantané de longueurs de mots l_i que nous avons choisies. Il reste à étudier sa longueur moyenne. On a :

$$\bar{L} = \sum_{i=1}^{n} p_i l_i < -\sum_{i=1}^{n} p_i \frac{\log_2 p_i}{\log_2 d} + 1 = -\frac{1}{\log_2 d} \sum_{i=1}^{n} p_i \log_2 p_i + 1 = \frac{H(S)}{\log_2(d)} + 1$$

et

$$\bar{L} = \sum_{i=1}^{n} p_i l_i \geq -\sum_{i=1}^{n} p_i \frac{\log_2 p_i}{\log_2 d} = \frac{H(S)}{\log_2(d)}$$

C.Q.F.D

2.2.3 Extension de source et le premier théorème de Shannon

Nous avons déjà vu que, pour un alphabet et une distribution de probabilité donnés, il est possible de trouver un code déchiffrable est proche de la borne inférieure $\frac{H(S)}{\log_2(d)}$ à un bit de base d près. La dernière question qu'il reste à éclaircir est de savoir s'il est possible d'approcher cette borne avec une précision arbitraire.

Pour répondre à cette question nous allons élargir notre point de vue sur le problème d'encodage de messages d'une source. Nous allons introduire l'idée de codage par blocks. Commençons par un exemple.

Exemple 3.2. Soit une source X d'alphabet $\omega_X = \{a, b\}$ et de distribution de probabilité $P_X = \{3/4, 1/4\}$. L'entropie de cette source est

$$H(X) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = -\frac{3}{4}\log_2(3) + \frac{3}{4}\log_2(4) + \frac{1}{4}\log_2(4) = -\frac{3}{4}\log_2(3) + 2 \simeq 0.811$$

En codant cette source caractère par caractère, on peut envisager le code suivant : $m(a) = 0,\ m(b) = 1$ de longueur moyenne de mots $\bar{L}_1 = 1$ bit par caractère. Un message T de longueur initiale $l(T)$ sera alors codé par une suite binaire de $l(T)$ bits.

Et si on voulait coder les messages de cette source en associant un code au couples de caractères ? Tout d'abord, formalisons cette idée. Avec un alphabet de taille 2 il est possible de former $2^2 = 4$ couples de caractères différents. On peut considérer cela comme un nouvel alphabet, d'une nouvelle source, Y. La variable aléatoire Y associée à cette nouvelle source correspond à l'observation de deux caractères émis indépendamment par la source initiale X. Ainsi, $Y = (X_1, X_2)$ où X_1 et X_2 sont deux variables aléatoires indépendantes ayant la même distribution que X. Compte tenu de l'indépendance de X_1 et X_2 on peut construire la distribution de probabilité de Y :

Y	aa	ab	ba	bb
P_Y	$p(a)^2 = 9/16$	$p(a)p(b) = 3/16$	$p(a)p(b) = 3/16$	$p(b)^2 = 1/16$

Enfin, l'entropie de Y peut aussi être déduite de l'indépendance de X_1 et X_2 :

$$H(Y) = H(X_1, X_2) = H(X_1) + H(X_2) = 2H(X)$$

Maintenant, si l'on considère Y comme une source à part entière, on peut appliquer le théorème 3.4. Il est possible de trouver un code déchiffrable dont la longueur moyenne de mots de code vérifie (ici $d = 2$ et donc $\log_2(d) = 1$) :

$$H(Y) \leq \bar{L}_2 =< H(Y) + 1 \quad \Leftrightarrow \quad 2H(X) \leq \bar{L}_2 =< 2H(X) + 1$$

On a alors la relation :

$$H(X) \leq \frac{\bar{L}_2}{2} =< H(X) + \frac{1}{2}$$

Il reste à remarquer que $\bar{L}_2$ est mesurée en "bits par symbole" de l'alphabet de Y. Or, tout symbole de l'alphabet de Y est un couple de caractères de l'alphabet de X. Ainsi, la quantité $\frac{\bar{L}_2}{2}$

représente la longueur moyenne de mot de code par symbole de X. Nous avons alors, un code dont la longueur moyenne de mot de code se trouve dans un intervalle plus petit (de longueur de 1/2 au lieu de 1) autour de la borne inférieure.

Voici l'exemple d'un tel code

$Y = (X_1, X_2)$	Proba	Code
aa	9/16	0
ab	3/16	10
ba	3/16	110
bb	1/16	111

et on a $H(Y) = 2H(X) \simeq 1.622$ et $\bar{L}_2 = \dfrac{27}{17}$ bits par valeur de Y.

Enfin, il est très important de remarquer que la longueur moyenne $\bar{L}_2$ représente le nombre moyen de bits pour coder un "symbole" de l'alphabet de la source Y. Donc il s'agit de $\dfrac{27}{17}$ bits par symbole de Y. or, chaque symbole de l'alphabet de Y est un couple de symboles de l'alphabet de X. Ainsi, en unités de mesure de la source X on a $\bar{L}_2 = \dfrac{27}{2 \cdot 17} \simeq 0.8$ bits par symbole de X.

Cela signifie qu'un message T de longueur $l(T)$ sera codé avec le code 2, en moyenne par $0.8l(T)$ bits binaires au lieu de $l(T)$ bits avec le premier code.

Ainsi, en introduisant le codage par blocks nous avons trouvé la possibilité de s'approcher de la borne inférieure. En généralisant l'idée de l'exemple précédent on peut formuler et démontrer le premier théorème fondamental de la théorie de l'information.

Définition 3.1 (Extension de source)**.** Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, n\}$. On appelle extension d'ordre s de la source X la source $Y = (X_1, \ldots, X_s)$ où $X_i,\ i = 1, \ldots, s$ sont les variables aléatoires indépendantes et identiquement distribuées selon la distribution de X.

Théorème 3.5 (Premier théorème de Shannon)**.** *Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ de taille n et de distribution de probabilités $P_X = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Alors il existe un procédé de codage déchiffrable dont la longueur moyenne de mots de code est aussi voisine que l'on souhaite de la borne inférieure $\dfrac{H(S)}{\log_2(d)}$.*

Preuve du théorème 3.5

Soit la suite $(Y_s)_{s=1}^{\infty}$ d'extensions d'ordre s de la source Y. On alors $\forall\ s \geq 1,\ \ H(Y_s) = sH(X)$ et il existe un code déchiffrable dont la longueur moyenne $\bar{L}_s$ de mots de code vérifie :

$$\frac{H(Y)}{\log_2(d)} \leq \bar{L}_s =< \frac{H(Y)}{\log_2(d)} + 1 \quad \Leftrightarrow \quad s\frac{H(X)}{\log_2(d)} \leq \bar{L}_s =< s\frac{H(X)}{\log_2(d)} + 1$$

On a alors la relation :

$$\frac{H(X)}{\log_2(d)} \leq \frac{\bar{L}_s}{s} =< \frac{H(X)}{\log_2(d)} + \frac{1}{s}$$

Il reste à remarquer que $\bar{L}_s$ est mesurée en "bits par symbole" de l'alphabet de Y_s. Or, tout symbole de l'alphabet de Y_s est un s-uplet de caractères de l'alphabet de X. Ainsi, la quantité $\frac{\bar{L}_s}{s}$ représente la longueur moyenne de mot de code par symbole de X.
Alors en passant à la limite quand s tend vers ∞ on a

$$\lim_{s\to\infty} \frac{\bar{L}_s}{s} = \frac{H(X)}{\log_2(d)}$$

C.Q.F.D

3 Construction de codes optimaux

Le premier théorème de Shannon, vu dans le chapitre précédent établit l'*existence* de codes dont la longueur moyenne de mots est aussi proche de la borne inférieure que l'on veut. Il ne propose cependant pas de méthode pour construire un tel code.

Rappelons qu'un code est dit absolument optimal pour une source donnée (représentée par son alphabet et sa distribution de probabilité) si sa longueur moyenne de mots atteint la borne inférieure. Nous avons mentionné dans le chapitre précédent que tels codes n'existent pas toujours pour une source donnée.

Définition 3.2 (Code optimal)**.** Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, n\}$ et de distribution de probabilité P_X donnés. On dit qu'un code est **optimal** dans une classe de codes pour cette source si sa longueur de mots de code est minimale dans la classe considérée.

Dans ce chapitre nous nous intéressons au problème de construction effective de **codes optimaux**. Pour faciliter la compréhension, nous allons nous restreindre à l'étude de **codes binaires sans préfixe** (instantanés). La généralisation des idées que nous allons exposer ici au cas d'alphabets de codes q-aires (de taille q) n'est pas difficile. De plus, les codes binaires occupent une grande place parmi les codes actuellement utilisés tout simplement à cause des principes de fonctionnement des ordinateurs.

3.1 Codes binaires instantanés et arbres

3.1.1 Quelques rappels sur les arbres

Nous rappelons ici quelques notions utiles dans la suite sur les arbres. Etant donné que nous avons choisi de présenter dans ce chapitre seulement les codes binaires, nous n'aurons besoin que

d'*arbres binaires*. On peut consulter pour plus de détails le cours sur les graphes de Marietta Manolessou dans la matière "Algorithmique II".

Un arbre binaire est un cas particulier de graphe qui peut être facilement défini de façon recursive : *un arbre binaire est soit vide soit composé d'un nœud particulier, appelé racine, d'un sous-arbre gauche et d'un sous-arbre droit qui sont eux mêmes des arbres binaires disjoints.* Nous allons tirer de cette définition quelques propriétés essentielles et ne description plus détaillée de la structure d'un arbre binaire.

1. Comme tout graphe, un arbre binaire est un couple (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs reliant certains sommets. Un arbre binaire est graphe orienté dans lequel les arcs sont considérés comme des relations "père-fils".
2. La **racine de l'arbre** est l'unique nœud qui n'a pas de père.
3. Dans un arbre binaire, chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
4. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
5. les nœuds qui ne sont ni feuilles ni racine s'appellent internes.
6. Un chemin entre deux nœuds est une suite d'arcs consécutifs (deux arcs (i, j) et (k, l) sont consécutifs si l'extrémité du premier est l'origine de l'autre : $j = k$). La longueur d'un chemin est le nombre de branches qui le constituent.
7. Tout nœud est relié à la racine par un unique **chemin**. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
8. On appelle **hauteur** ou **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
9. Un **arbre binaire complet de profondeur** n est un arbre dont tous les nœuds sauf les feuilles ont exactement 2 fils. Autrement dit, toutes les feuilles appartiennent au même niveau et tous les nœuds ont exactement 2 ou 0 fils. Un tel arbre a exactement 2^n feuilles et pour tout $i \leq n$ il y a exactement 2 nœuds de niveau i.
10. On appelle **arbre binaire incomplet** un arbre binaire obtenu à partir d'un arbre complet en supprimant tous les successeurs d'un certain nombre de nœuds ainsi que toutes les branches qui touchent ces nœuds. Les nœuds dont a supprimé les successeurs deviennent les feuilles. Attention ! Il existe des arbres qui ne sont ni complets, ni incomplet. Un arbre incomplet est un arbre dont tous les nœuds ont 2 ou 0 fils. A la différence d'un arbre complet, il n'est pas exigé que toutes les feuilles soient de même niveau. **Attention !** Il existe des arbres qui ne sont ni complets ni incomplets.
11. Dans un arbre binaire complet ou incomplet le nombre de feuilles F et le nombre de branches B vérifient la relation
$$B = 2(F - 1)$$

Sur la figure 4.1 on peut voir un arbre binaire complet de profondeur 3 et plus loin sur la figure 4.2un arbre incomplet obtenu à partir du premier en supprimant les descendants de deux noeuds.

3.1.2 Représentation de codes instantanés par les arbres

Soit $\{m_1, \ldots, m_n\}$ un code binaire sans préfixes de longueur maximale l. Il est évident que chaque mot de ce code peut être représenté par un chemin partant de la racine d'un arbre binaire

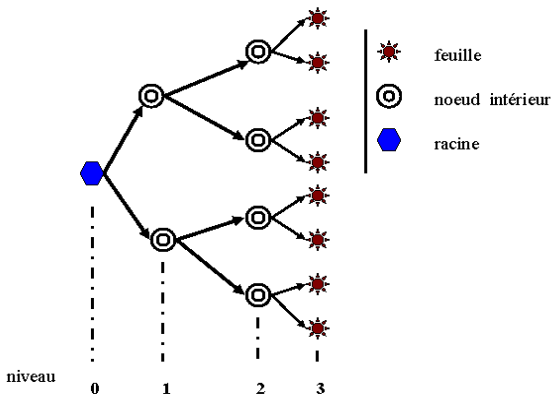

FIGURE 3.1 – Un arbre binaire complet de profondeur 3.

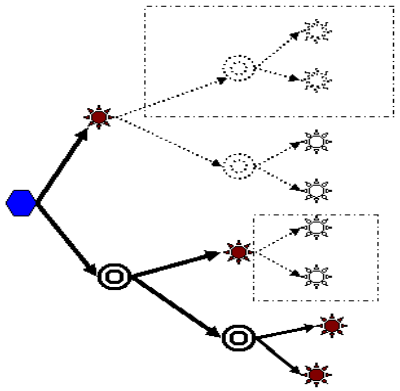

FIGURE 3.2 – Un arbre binaire incomplet

complet de profondeur l. il suffit pour cela d'étiqueter les arcs du l'arbre avec 0 et 1. Supposons qu'à un mot m_i de longueur $l_i \leq l$ on vient d'associer un chemin de l_i arcs en partant de la racine. Le chemin s'arrête alors à un noeud de niveau l_i. Comme aucun autre mot du code ne peut avoir celui ci comme préfixe, on peut supprimer tous les descendants du noeud final du chemin. Ce dernier devient alors une feuille. Ainsi on peut associer à tout code sans préfixes un arbre binaire incomplet. L'arbre vu précédemment, est ainsi associé au code $\{1, 01, 001, 000\}$ (voir la figure **??**).

Ainsi dans un arbre correspondant à un code binaire, les feuilles correspondent aux mots du code. Si ce dernier est associé à l'alphabet d'une source, il est également possible d'associer aux feuilles les probabilités des symboles correspondants.

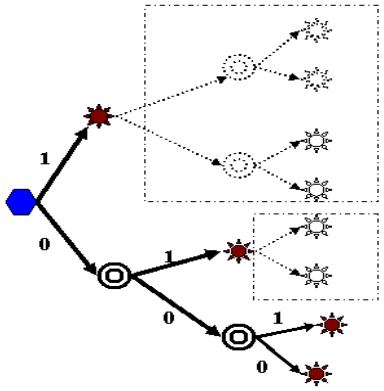

FIGURE 3.3 – Représentation d'un code par un arbre binaire incomplet

4 Méthode de Huffman de construction de codes optimaux

Nous allons maintenant présenter la méthode de Huffman de construction de codes instantanés optimaux pour une source donnée. Nous commençons par condition nécessaire d'optimalité que nous présentons ici sans démonstration mais dans le but de faciliter la compréhension de la procédure de Huffman.

Lemme 3.1. *(Condition nécessaire d'optimalité) Soit une source X représentée d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ et de distribution de probabilités $P_X = \{p_1, \ldots, p_n\}$. Soit $C = \{m_1, \ldots, m_n\}$ un code instantané associé à la source de longueurs de mots $\{l_1, \ldots, l_n\}$. Supposons que les symboles de l'alphabet sont numérotés dans l'ordre décroissant de leurs probabilités :*

$$p_1 \geq p_2 \geq \cdots \geq p_n.$$

Si plusieurs symboles ont la même probabilité, ils sont numérotés dans l'ordre croissant de longueurs de mots de code correspondants.

Si C est optimal pour la source X dans la classe de codes instantanés alors il vérifie les propriétés suivantes :

1. *Les symboles les plus probables ont les longueurs de mots de code plus petites :*

$$p_i \geq p_j \quad \Rightarrow \quad l_i \leq l_j$$

2. *Les deux derniers symboles ont les longueurs de mots de code égales :*

$$l_n = l_{n-1}$$

3. *Parmi les symboles dont la longueur de mots de code est l_n il y a au moins deux dont les mots de code ont tous les chiffres identiques sauf le dernier.*

La méthode de Huffman (1952) procède de façon recursive, en construisant l'arbre binaire du code à partir des feuilles du niveau le plus élevé. D'après le lemme ci-dessus ces feuilles doivent correspondre aux symboles les moins probables x_n et x_{n-1}. Le principe général de la construction est de regrouper les deux symboles les moins probables en un seul, $w_{n,n-1}$ en additionnant leurs probabilités et de considérer une nouvelle source de $n-1$ symboles. On parle alors de réduction de source. On applique ainsi le principe récursivement jusqu'à ce qu'il ne reste que deux symboles. On leur attribue alors le code $\{0, 1\}$.

Le parcours inverse permet de construire le code de manière également recursive. Soit C_{n-1} le code associé à la source de $n-1$ symboles dont le dernier est $w_{n,n-1}$ a été obtenu en regroupant x_n et x_{n-1}. Alors le code C_n associé à la source de n symboles est obtenu de façon suivante. Les mots du code associés aux symboles x_n et x_{n-1} sont construits en ajoutant au mot code du symbole $w_{n,n-1}$ respectivement 0 et 1.

Voici un exemple.

Etape 1		Etape 2		Etape 3		Etape 4		Etape 5	
x_1	0.3	x_1	0.3	x_1	0.3	$x_{3,4,5,6}$	0.45	$x_{3,4,5,6}$	0.45
x_2	0.25	x_2	0.25	x_2	0.25	x_1	0.3	$x_{1,2}$	0.55
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25	x_2	0.25		
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								

L'arbre représentant le code résultant est représenté sur la figure suivante :

On y retrouve les codes associés à tous les symboles de l'alphabet initial.

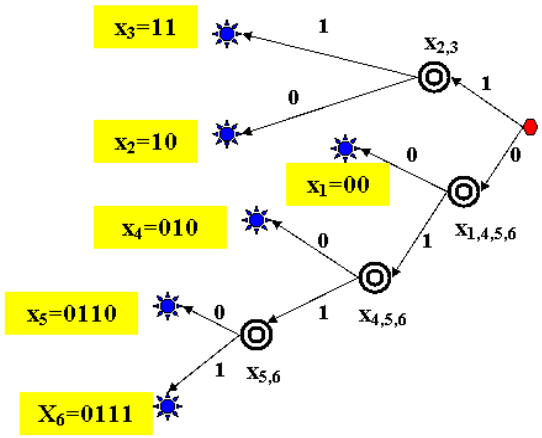

FIGURE 3.4 – Arbre de code de Huffman

Chapitre 4

Compression de données sans pertes

1 Codage et compression

Nous allons dans ce chapitre aborder les applications de la théorie de l'information à la compression des données numériques. Nous avons décrit dans le chapitre précédent les méthodes de codage qui permettent d'adapter les messages émis par une source à la transmission via un canal tout en assurant la transmission la plus rapide en moyenne et un décodage sans erreurs ni ambiguïtés. Nous avons associé la rapidité de transmission à la longueur des messages codés. Le paramètre clé est alors la longueur moyenne des mots de code.

Les résultats fondamentaux de codage de source ont une autre application : la compression des données. En effet, le premier théorème de Shannon permet d'établir les limites théoriques de compression de données sans pertes. En effet, l'entropie d'un message représente la longueur moyenne minimale de mots de code pour ce message. Le codage à longueur de mots fixe est souvent utilisé car il représente l'avantage de simplicité et rapidité d'application. Il est tout même loin d'être optimal et lorsque les données obtenues sont trop volumineuses il se pose le problème de compression.

Il existe deux grandes familles de méthodes de compression :

Compression sans pertes. Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

Compression avec pertes. Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats Jpeg, MP3, MPEG.

Dans ce cours nous allons nous intéresser uniquement aux méthodes de compression sans pertes, basées sur l'utilisation des algorithmes de codage plus ou moins proches de la borne inférieure établie par le premier théorème de Shannon.

On peut regrouper les algorithmes de codage pour compression en deux familles principales :

Algorithmes statistiques. Il s'agit de méthodes basées sur l'utilisation de la table de fréquences d'occurrence des symboles, comme le codage de Huffman.

Algorithmes à dictionnaire. Il s'agit de coder non pas par caractère par caractère mais par blocks en indiquant la position de chaque bloc dans un dictionnaire construit à partir du message.

Tout d'abord, nous introduisons quelques notions clés qui nous seront utiles pour apprécier les qualités de différents codes en tant que méthodes de compression des données. Nous avons vu dans le chapitre 2 qu'il existe une borne inférieure pour la longueur moyenne de mots de tout code déchiffrable dont l'alphabet de canal a d caractères :

$$\frac{H(S)}{d}$$

où $H(S)$ est l'entropie de la source. Dans ce chapitre, nous allons nous restreindre à l'étude de codes binaires, c'est à dire, avec $d = 2$ et donc $\log_2(d) = 1$. Ainsi, dans ce cas particulier, c'est l'entropie de la source $h(S)$ qui représente la valeur minimale possible de longueur moyenne de mots de code.

Soit une source S d'alphabet $\Omega = \{s_1, \ldots, s_n\}$ et d'entropie $H(S)$. Soit un code binaire déchiffrable pour S définit par l'ensemble de mots $\{w_1, \ldots, w_n\}$ de longueurs respectives $\{l_1, \ldots, l_n\}$. Soit $\bar{L}$ la longueur moyenne de mots de code. On appelle **efficacité** du code la quantité

$$E = \frac{H(S)}{\bar{L}}$$

et **redondance** du code la quantité

$$R = 1 - E = 1 - \frac{H(S)}{\bar{L}}$$

Ces quantités permettent de comparer la longueur moyenne de mot de code d'un code donné à la borne inférieure $h(S)$. Rappelons que cette borne n'est pas toujours atteinte, même par les codes optimaux.

Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$. Soit M le nombre de symboles dans T. Si le texte est codé à l'aide d'un code binaire de longueur de mots fixe (ASCII ou UTF-8, par exemple) la mémoire occupé par ces données est proportionnelle au nombre de symboles dans le texte. Par exemple, si on utilise R bits par symbole, la longueur totale de texte codé est $l(T) = RM$. Le problème de compression des données dans ce cas consiste à rechercher un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$. On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

Pour appliquer à ce problème la théorie de construction de codes optimaux vue dans le chapitre précédent, nous allons définir la procédure permettant d'associer à un texte donné une distribution de probabilité. pour tout symbole s_i de l'alphabet Ω soit m_i le nombre d'occurrences de ce symbole dans le texte T. Notons alors $f_i = \dfrac{m_i}{M}$ la fréquence du symbole s_i. On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

Ainsi, l'ensemble $\{f_i,\ i = 1, \ldots, n\}$ est une distribution de probabilité sur Ω.

Cela nous permet de considérer le texte à compresser T comme étant un message émis par une source ayant cette distribution de probabilité. Alors les méthodes de construction de codes optimaux et en particulier la méthode de Huffman peuvent être appliquées pour la compression de ce texte.

Il est important de souligner que dans ce cas le code construit est propre au texte T et le taux de compression obtenu dépend également des caractéristiques du texte.

2 Algorithmes de compression statistiques

2.1 Méthode de Shannon-Fano

On suppose la table de fréquences d'apparition des symboles $\{f_i,\ i = 1, \ldots, n\}$ construite. La méthode de Shannon-Fano est une méthode recursive. Elle permet de construire l'arbre du code en partant de sa racine, contrairement à la méthode de Huffman qui procède à partir des feuilles.

Les étapes de la méthode sont les suivantes :

1. On élimine de l'alphabet Ω les symboles dont les fréquences sont nulles et on trie la table dans l'ordre décroissant des fréquences.
2. On divise la table de fréquences en deux parties de telle sorte que les sommes des fréquences de ces parties soient aussi proches que possible. On attribue 0 à la première partie et 1 à la seconde.
3. On applique ensuite ce principe des divisions à chacune des parties recursivement jusqu'à l'obtention de parties à un seul symbole.

2.1.1 Exemple

Soit T="cette échelle est belle". Il y a $M = 20$ symboles dans ce texte. Codé en ASCII (4 bits par symbole) il occupe alors $l(T) = 4 \cdot 20 = 80$ bits. Nous retenons cette valeur comme référence pour évaluer les taux de compression obtenus. Nous allons construire pour ce texte les codes de Shannon-Fano et de Huffman. Tout d'abords, on construit la table des fréquences d'occurrence des symboles du texte :

s_i	e	l	t	c	b	h	s
m_i	8	4	3	2	1	1	1
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

Construction du code de Shannon-Fano. Pour commencer on divise l'alphabet en deux sous-ensembles de sommes de probabilités aussi proches que possible. Ici on a $\Omega_1 = \{e, l\}$,

$P(\Omega_1) = 0.4 + 0.2 = 0.6$ et $\Omega_2 = \{t, c, b, h, s\}$, $P(\Omega_2) = 0.4$. On associe 1 comme premier bit des mots de code des symboles de Ω_1 et 0 comme premier bit des mots de code des symboles de Ω_2. Sur chacun des sous-alphabets ainsi obtenus on définit les distributions de probabilités en divisant celles de Ω par $P(\Omega_1)$ et $P(\Omega_2)$ respectivement. On obtient :

$P_1 =$

s_i	e	l
f_i	4/6	2/6

$P_2 =$

s_i	t	c	b	h	s
f_i	3/8	1/4	1/8	1/8	1/8

On continue ensuite, en appliquant la même procédure aux deux tableaux ci-dessous. Voici le graphe ainsi obtenu :

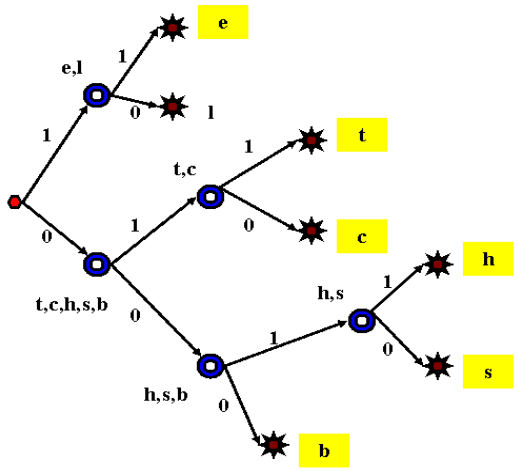

FIGURE 4.1 – Codage de Shannon-Fano

Le code qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
w_i	11	10	011	010	000	0011	0010
l_i	2	2	3	3	3	4	4

La longueur moyenne de mots de code est $\bar{L} = 2.5$. Le message codé sera (sans les espaces)

$$U = 01011011011111101000111110101111001001100011101011$$

et on a $l(U) = 50$ Ainsi le taux de compression est

$$t = \frac{50}{80} \simeq 0.625$$

2.2 Algorithme de Huffman

Le codage de Huffman représente aussi une méthode de compression car il réduit la longueur moyenne des mots de code.

Nous avons décrit cette méthode en détails dans les chapitres précédents. Voici un exemple d'utilisation dans le cas vu dans la section précédente.

Construction du code de Huffman. En utilisant la méthode de Huffman (sans détails ici) on obtient l'arbre de code suivant :

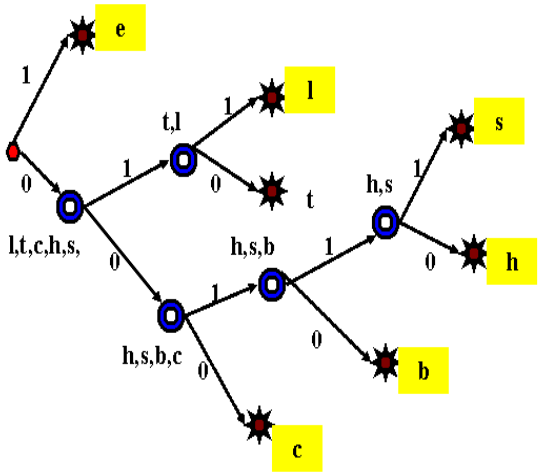

FIGURE 4.2 – Codage de Huffman

Le code qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
w_i	1	011	010	000	0010	00110	00111
l_i	1	3	3	3	4	5	5

La longueur moyenne de mots de code est $\bar{L} = 2.45$. Le message codé sera (sans les espaces)

$$U = 000101001011000001101011011110011101000101010110111$$

et on a $l(U) = 49$ Ainsi le taux de compression est

$$t = \frac{49}{80} \simeq 0.6125$$

3 Codes à dictionnaire

Les trois algorithmes les plus connus sont LZ77(Lempel-Ziv), LZ78(Lempel-Ziv) et LZW (Lempel-Ziv-Welsh, 1984). Le principe fondateur commun des ces trois méthodes consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire. Le dictionnaire est construit à partir du texte lui même et contient toutes les séquences déjà rencontrées. Nous allons dans la suite détailler le fonctionnement de l'algorithme LZ78.

3.1 LZ78

3.1.1 Codage

Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser. A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées. Le numéro 0 est réservé à la chaîne de caractères vide.

Chaque séquence codée sera remplacée par un couple : (i, c). Si la séquence codée a n caractères, i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères. Si aucun préfixe n'a été trouvé, $i = 0$. c est le dernier caractère de la séquence codée.

Pour déterminer la nouvelle séquence à encoder on procède de la façon suivante. On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence s est de la forme

$$s = s_d c$$

où c est le dernier caractère lu et s_d est la dernière séquence rencontrée qui appartient au dictionnaire.

Illustrons le propos par un exemple. Supposons que nous cherchons à compresser le texte "ELLE EST BELLE CETTE ECHELLE ETERNELLE.".

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,' E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,' L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$. On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.
- Caractère lu : $\sqcup$. Il n'existe pas dans le dictionnaire. On le remplace par $(0, \sqcup)$ et on l'enregistre à l'adresse $i = 4$.
- caractère lu : "E", adresse dans le dictionnaire : $i = 1$. On lit la suivant. Séquence en attente : "ES". On la remplace par $(1, S)$ et on l'enregistre à l'adresse $i = 5$.
- etc

Voici le tableau qui résume toute la procédure :

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
4	␣	(0, ␣)	17	ETE	(11,E)
5	ES	(1,S)	18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4, B)	20	ELLE	(15,E)
8	EL	(1,L)			
9	LE␣	(3, ␣)			
10	C	(0,C)			
11	ET	(1,T)			
12	TE	(6,E)			

Le code définitif obtenu sera

$$(0E, 0L, 2E, 0␣, 1S, 0T, 4B, 1L, 3␣, 0C, 1T, 6E, 4E, 10H, 8L, 1␣, 11E, 0R, 0N, 15E)$$

Si l'on utilise un octet pour coder une adresse de dictionnaire (nombre entier) et un octet pour coder un caractère, nous arrivons ainsi à 40 caractères pour le code final au lieu de 38 pour le texte initial. Il peut sembler que ce codage n'est pas efficace pour compresser les données, bien au contraire !

Son efficacité s'améliore nettement sur des textes longs. Voici notre exemple, juste un peu prolongé : "ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.".

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(O,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
4	␣	(0, ␣)	17	ETE	(11,E)
5	ES	(1,S)	18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4, B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3, ␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)
12	TE	(6,E)	25	ELLE.	(20,.)

On obtient un code de 50 caractères au lieu de 56.

Nous allons maintenant énoncer (sans démonstration) un certain nombre de résultats sur les qualités de compression obtenues en fonction de la longueur du texte. Soit un texte de longueur n produit par une source S. Alors la taille maximale du dictionnaire $d(n)$ vérifie la relation

$$d(n) \leq \frac{n}{(1-\delta_n)\log_2(n)}, \quad \delta_n \to 0,\ n \to \infty$$

Ainsi la croissance du dictionnaire est moins que linéaire. Si le dictionnaire a $d(n)$ mots, on utilisera en moyenne $\log_2(c(n))+1$ bits par mot pour un codage binaire. Ainsi le nombre de bits moyen par symbole du texte original est

$$\frac{c(n)}{n}(\log_2(c(n))+1)$$

Enfin, on peut établir que le codage de Lempel-Ziv est assyptotiquement optimal (au sens du premier théorème de Shannon)

$$\lim_{n\to\infty} \frac{c(n)}{n}(\log_2(c(n))+1) = H(S)$$

3.1.2 Décodage

L'avantage de la méthode de Lempel-Ziv est que le dictionnaire n'a pas besoin d'être transmis. Seul le code est envoyé. Le dictionnaire, nécessaire au décodage, sera reconstruit par le décodeur. A chaque couple de code lu le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple ;
2. Enregistrer la nouvelle séquence dans le dictionnaire.

Voici le décodage du code de notre premier exemple :

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N, 15E)$$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'
- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '$\sqcup$B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, \sqcup)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '

- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

3.2 LZW

L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv. Elle permet de réduire le code en remplaçant le couple (i, c) par le seul indice i du dictionnaire.

Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout. On lit ensuite le texte à compresser à travers une fenêtre dont la taille grandit, caractère par caractère, jusqu'à rencontrer une séquence qui n'est pas dans le dictionnaire. Comme dans l'algorithme LZ78, cette séquence, si elle est de longueur l, est composée d'un mot du dictionnaire de longueur $l-1$ et d'un dernier caractère lu. Au lieu de coder cette séquence par le couple (i, c), comme dans LZ78, on code le mot déjà existant dans le dictionnaire, en le remplaçant par son adresse i. On ajoute ensuite la nouvelle au dictionnaire cette nouvelle séquence à une nouvelle adresse. La prochaine fois qu'elle sera lue dans le texte elle sera codée par son adresse.

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"EL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 256$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 257$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LE". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 258$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"E␣". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 259$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère "␣". Cette séquence est présente dans le dictionnaire, $i = 32$. On lit le caractère suivant :"␣E". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 260$. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"ES". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 261$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc

Voici le tableau qui résume toute la procédure :

indice	Dictionnaire	Code	indice	Dictionnaire	Code
256	EL	69	269	ET	69
257	LL	76	270	TT	84
258	LE	76	271	TE	84
259	E␣	69	272	E␣E	259
260	␣E	32	272	EC	69
261	ES	69	273	CH	67
262	ST	83	274	HE	72
263	T␣	84	275	ELL	256
264	␣B	32	276	LE␣	258
265	BE	66	277	␣ET	260
266	LLE	257	278	TER	271
267	E␣C	259	279	RN	82
268	CE	67	280	NE	78
			281	ELLE	275

Le code obtenu pour la phrase sera

$$69, 76, 76, 69, 32, 69, 83, 84, 32, 66, 257, 259, 67, 69, 84, 84, 259, 69, 67, 72, 256, 258, 260, 271, 82, 78, 275, 69$$

Nous avons 28 nombres entre 0 et 512 qui peuvent être codés sur 9 bits. On a ainsi 252 bits de code. le texte initial a 38 caractères ASCII, codés sur 8 bits, donc 304 bits. Bien évidemment, le taux de compression est meilleur pour un texte grand.

Le décodage se fait, comme dans le cas de LZ78 de manière inverse, en restituant le disctionnaire au fur et mesure de la lecture du code.

3.3 Remarques

La formulation de l'algorithme ne dit rien de particulier sur la taille du dictionnaire. En théorie, l'algorithme fonctionne avec un dictionnaire illimité. Dans la pratique, il est toujours de taille fixe. Lorsqu'il est plein on a plusieurs choix(selon la méthode utilisée) : soit de l'étendre, soit d'en "oublier" une partie, soit de se contenter de l'utiliser tel quel pour encoder simplement le reste du texte.

Le taux de compression obtenu peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses. On exploite ainsi les fréquences d'occurrence non seulement des caractères mais aussi des séquences de taille variable.

De nombreux utilitaires de compression des données utilisent les algorithmes à dictionnaire.

La commande compress sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots (codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille (codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.

GZIP, PKZIP, WINZIP Ces logiciels utilisent une variante de l'algorithme LZ77

Formats d'images GIF, PNG utilisent également une variante de codage par dictionnaire.

4 Exemple. Compression d'images.

Les techniques de compression d'images sont aujourd'hui nombreuses, tant dans le groupe de méthodes avec pertes (JPEG) que dans celui des méthodes sans pertes (GIF).

Nous allons aborder ici quelques méthodes simples de compression sans pertes utilisées par les différents formats d'images disponibles.

Pour simplifier les considérations ci-dessous, nous allons étudier les images dites monochromes (en niveaux de gris, par exemple). Une telle image peut être représentée comme une matrice contenant les valeurs codées d'intensités lumineuses de chaque point numérique (pixel). La profondeur de codage est le nombre de bits utilisés pour un seul pixel, généralement, 8 ou 16. Nous allons considérer que les valeurs sont représentées directement. Il existe aussi une approche par palette, une sorte de dictionnaire de couleurs, utilisée pour les images de synthèse. Dans le cas des images réelles, l'utilisation des palettes de couleur est possible mais elle induit une compression avec pertes.

4.1 Codage RLE (Run-length Encoding)

Cette méthode est utilisée par les formats BITMAP et GIFF. Sa popularité est surtout due à la simplicité de sa mise en pratique. Cette méthode de codage n'est pas spécifique aux images : elle est applicable à n'importe quel type de données. Mais son efficacité dépend beaucoup de la nature des données à compresser. Pour simplifier les explications, on le présente d'abord comme méthode de compression de texte.

On recherche dans le texte à compresser des séquences de caractères répétés et on les remplace par des couples (N, c) où N est le nombre de répétitions et c est le caractère à répéter. Par exemple, la séquence

$$eeeeeeee$$

sera codée par $(8e)$. Alors le texte $tttaaattaarrrrbbb$ sera codé comme : $3t3a2t2a4r3b$. Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII. Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères. Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère. Ainsi la phrase "ma jolie phrase" aura un code RLE deux fois plus long que le code ASCII car aucun caractère ne se répète de façon succéssive.

Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs. Par exemple, une image de damier en noir et blanc, aura un taux de compression très élevé. Le codage se fait alors ligne par ligne, en partant du coin en haut à gauche.

4.2 Codage différentiel

Cette technique est souvent utilisée comme traitement avant une méthode de compression. On suppose que dans une image les valeurs de pixels sont liées et qu'il est possible de prédire, approximativement, la valeur d'un pixel à partir de ses voisins déjà traités. Dans un parcours

d'une image ligne par ligne, en commençant par le coin en haut à gauche, pour le pixel courent on connaît les valeurs de ses voisins de la ligne au-dessus (HG(haut-gauche), HC(haut-centre) et HD(haut-droite)). Dans la même ligne on connaît son précédent, P. Alors, en prenant en compte tous ou une partie de ces voisins, on peut estimer la valeur du pixel à l'aide de la moyenne des voisins. Il est donc inutile de coder cette information, puisque elle peut être restituée. Il suffit de coder juste l'erreur de prédiction : la différence entre la valeur prédite et la valeur réelle du pixel.

Généralement, on obtient de très bons taux de compression, en couplant le codage différentiel avec une méthode de compression par dictionnaire. C'est le cas pour le format PNG, par exemple.

4.3 Méthodes mixtes

De nombreux formats de compression avec pertes utilisent le codage réversible (Huffman ou par dictionnaire) pour optimiser les taux de compression obtenus. Le format JPEG par exemple, utilise une transformée en cosinus discrets (DCT) dont les coefficients subissent une procédure de quantification, et ensuite applique un codage de Huffman pour réduire encore la place occupée par le code.

Chapitre 5

Modélisation mathématique d'un canal

1 Entropie conjointe - Entropie conditionnelle

2 Quelques notions utiles de probabilités

Soit X une variable aléatoire discrète à support fini définie sur un espace probabilisé fini. Une telle variable est définie par la donnée de son support (ensemble de valeurs possibles) $X \in \{x_i\}_{i=1}^n$ et par sa distribution de probabilité,

$$P : \{x_i\}_{i=1}^n \to [0,1], \quad x_i \mapsto P(x_i)$$

telle que $\forall i = 1, \ldots, n 0 \leq P(x_i) \leq 1$ et $\sum_{i=1}^{n} P(x_i) = 1$.

À un couple de variables aléatoires X à valeurs dans $\{x_i\}_{i=1}^n$ et Y à valeurs dans $\{y_j\}_{j=1}^m$ on associe **la distribution conjointe de probabilités**

$$P_{XY} : \{(x_i, y_j),\ (i,j) \in |[1,n]| \times |[1,m]|\} \to [0,1], \quad (x_i, y_j) \mapsto P(x_i, y_j) = P[X = x_i \text{ et } Y = y_j]$$

On a les propriétés suivantes :

$$\forall (i,j),\ 0 \leq P_{XY}(x_i, y_j) \leq 1$$
$$\sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) = 1$$

Étant donnée la distribution conjointe d'un couple de variables aléatoires (X, Y), $P(x_i, y_j)$, on définit les **distributions marginales** de X et de Y respectivement par les relations :

$$\begin{cases} P_X : \{x_i\}_{i=1}^n \to [0,1], \quad x_i \mapsto P_X(x_i) = \sum_{j=1}^m P(x_i, y_j) \\ \\ P_Y : \{x_j\}_{j=1}^m \to [0,1], \quad y_j \mapsto P_Y(y_j) = \sum_{i=1}^n P(x_i, y_j) \end{cases}$$

Soient X et Y deux variables aléatoires discrètes. On associe à l'événement $Y = y_j$ une distribution conditionnelle $P[X|y_j]$ de X sachant y_j définie par

$$P : x_i \mapsto P(x_i|y_j) = P[X = x_i|Y = y_j].$$

3 Définitions et propriétés essentielles

Définition 5.1 (Entropie conjointe)**.** Soient $X = \{x_i\}_{i=1}^n$ et $Y = \{y_j\}_{j=1}^m$ deux variables aléatoires discrètes définies sur un même univers.
Soit $P(i,j) = P[X = x_i \text{ et } Y = y_j]$ leur distribution conjointe.
Alors **l'entropie conjointe de X et Y** est définie par

$$H(X,Y) = -\sum_{i=1}^n \sum_{j=1}^n P(i,j) \log_2(P(i,j)). \tag{5.1}$$

Cette définition peut être généralisée à un nombre quelconque r de variables aléatoires $X_1, \ldots, X_2$ à travers leur distribution de probabilité conjointe : $P(i_1, i_2, \ldots, i_r) = P[X_{i_1} = x_{i_1} \text{et } \ldots X_{i_r} = x_{i_r}]$:

$$H(X_1, X_2, \ldots, X_r) = -\sum_{i_1=1}^{n_1} \ldots \sum_{i_r=1}^{n_r} P(i_1, i_2, \ldots, i_r) \log_2(P(i_1, i_2, \ldots, i_r)).$$

Cette notion, que nous introduisons ici dans un contexte volontairement plus général de variables aléatoires, peut être interprétée de différentes façons selon l'application. Voici quelques exemples.

Exemple 5.1 (Étude des associations de symboles d'une source)**.** Soit une source S d'alphabet $\Omega = \{a, e, o, c, p, r, t\}$. Supposons que tous les symboles soient équiprobables. L'entropie de cette source est alors

$$H(S) = \log_2(7) \simeq 2.80 \text{ bits par symbole.}$$

Pour coder chaque symbole d'un message on ne peut pas utiliser en moyenne moins de 2.80 bits par symbole.

On peut se poser la question de distribution de différentes associations possibles de symboles entre eux. Par exemple, on peut étudier les syllabes formées d'une consonne et d'une voyelle.

Soit X la variable aléatoire associée aux voyelles $\{a, e, o\}$ et Y la v.a. associée à l'émission de consonnes $\{c, p, r, t\}$. On s'intéresse aux probabilités d'apparition simultanée d'une de ces voyelles et d'une de ces consonnes dans une même syllabe. Supposons que ces probabilités sont données par le tableau suivant :

$Y \backslash X$	a	e	o	$P_Y(y)$
c	0.05	0.05	0.1	0.2
p	0.1	0.15	0.05	0.3
r	0.05	0.1	0.1	0.25
t	0.15	0.05	0.05	0.25
$P_X(x)$	0.35	0.35	0.3	

L'entropie conjointe de X et Y est alors

$$H(X,Y) = -(6 \cdot 0.05 \log_2(0.05) + 4 \cdot 0.1 \log_2(0.1) + 2 \cdot 0.15 \log_2(0.15)) \simeq 3.44$$

On peut interpréter cette quantité comme entropie d'une source $Z = (X, Y)$ dont l'alphabet est formé de couples de symboles $(consonne, voyelle)$. Nous remarquons alors que pour coder chaque couple il suffirait en moyenne 3.44 bits par couple au lieu de $2 \times 2.80 = 5.6$ bits, si l'on codait chaque symbole du couple séparément.

Exemple 5.2 (Etude d'un couple émetteur-récepteur)**.** Dans la suite de ce cours nous allons étudier en détail les problèmes de transmission de messages émis par une source d'information via un canal de transmission. Nous allons donc associer à la source, d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ une variable aléatoire X dont les valeurs sont les symboles émis, et au récepteur, d'alphabet $\Omega_Y = \{y_1, \ldots, y_m\}$, une variable aléatoire Y dont les valeurs sont les symboles reçus. Alors la distribution conjointe de X et Y décrit le canal de transmission et son entropie est une caractéristique importante du canal.

Définition 5.2 (Entropie conditionnelle moyenne)**.** Soient $X = \{x_i\}_{i=1}^n$ et $Y = \{y_j\}_{j=1}^m$ deux variables aléatoires discrètes définies sur un même univers.
Soit $P(i,j) = P[X = x_i \text{et } Y = y_j]$ leur distribution conjointe.
Posons $P(i|j) = P[X = x_i | Y = y_j] = \dfrac{P(i,j)}{P[Y = y_j]}$.
Alors l'**entropie conditionnelle moyenne** de X sachant Y est définie par

$$H(X|Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P(i,j) \log_2(P(i|j)). \tag{5.2}$$

Remarque 5.1. Il est utile d'expliciter le terme "entropie conditionnelle moyenne". Supposons que $Y = y_j$ et considérons la distribution conditionnelle $P[X|y_j]$ de X sachant $Y = y_j$ définie par

$$P[X = x_i|y_j] = P[X = x_i|Y = y_j],\ i = 1, \ldots, n$$

On lui associe l'entropie conditionnelle de X sachant y_j comme l'information conditionnelle moyenne de X sachant y_j :

$$H(X|y_j) = -\sum_{i=1}^{n} P(x_i|y_j) \log_2(P(x_i|y_j)) = -\sum_{i=1}^{n} \frac{P(x_i, y_j)}{P(y_j)} \log_2(P(x_i|y_j))$$

Alors l'entropie moyenne de X sachant Y est définie par

$$\begin{aligned} H(X|Y) &= \sum_{j=1}^{m} P(y_j) H(X|y_j) \\ &= -\sum_{j=1}^{m} P(y_j) \sum_{i=1}^{n} \frac{P(x_i, y_j)}{P(y_j)} \log_2(P(x_i|y_j) \\ &= -\sum_{i=1}^{n} \sum_{j=1}^{m} P(i, j) \log_2(P(i|j)) \end{aligned}$$

On appelle souvent $H(X)$ *l'entropie a priori* de X et $H(X|Y)$ *l'entropie (moyenne) a posteriori* de X.

Proposition 5.1 (Additivité)**.** *L'entropie conjointe de deux variables aléatoires X et Y est égale à la somme de l'entropie de l'une d'elles et à l'entropie conditionnelle moyenne de l'autre.*

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

Corollaire 5.1.

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z)$$

Proposition 5.2 (Règles de chaînage)**.** *Soient $(X_i)_{i=1}^r$ r variables aléatoires discrètes. Alors*

1. *La loi de probabilités conjointes de $(X_i)_{i=1}^r$ peut être développée comme suit :*

$$\begin{aligned} P(X_1,\ldots,X_r) &= P(X_1)\cdot P(X_2|X_1)\cdot P(X_3|X_2,X_1)\ldots P(X_r|X_{r-1},\ldots,X_1) \\ &= \prod_{k=1}^{r} P(X_k|X_{k-1},\ldots,X_1) \end{aligned} \tag{5.3}$$

2. *L'entropie conjointe de $(X_i)_{i=1}^r$ vérifie*

$$\begin{aligned} H(X_1,\ldots,X_r) &= H(X_1)+H(X_2|X_1)+\ldots H(X_r|X_{r-1},\ldots,X_1) \\ &= \sum_{k=1}^{r} H(X_k|X_{k-1},\ldots,X_1) \end{aligned} \tag{5.4}$$

Proposition 5.3 (Propriétés)**.**

1. $$H(X,Y)\geq 0,\quad H(X|Y)\geq 0$$

   L'entropie conjointe $H(X,Y)$ est nulle ssi une seule des combinaisons (x_i,y_j) est possible. L'entropie conditionnelle moyenne $H(X,Y)$ est nulle ssi X est une fonction de Y.

2. $$H(X,Y)\geq \max(H(X),H(Y))$$

3. $$H(X,Y)\leq H(X)+H(Y)\leq 2H(X+Y)$$

4 Information mutuelle moyenne

Définition 5.3 (Information mutuelle moyenne)**.** Soient $X=\{x_i\}_{i=1}^n$ et $Y=\{y_j\}_{j=1}^m$ deux variables aléatoires discrètes définies sur un même univers. Soit $P(i,j)=P[X=x_i\text{et } Y=y_j]$ leur distribution conjointe. L'information mutuelle moyenne de X et Y est définie par

$$I(X;Y)=\sum_{i=1}^{n}\sum_{j=1}^{m} P(i,j)\log_2\left(\frac{P(i,j)}{P(x_i)P(y_j)}\right). \tag{5.5}$$

Remarque 5.2. À partir des définitions données ci-dessus on déduit facilement ces relations importantes :

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y) \tag{5.6}$$

Dans le cas où les variables X et Y représentent respectivement le message émis et le message reçu par le destinataire, cette relation signifie que l'information mutuelle moyenne est égale à :

- l'information émise $H(X)$, diminuée de l'incertitude sur le symbole x émis quand le symbole y reçu est connu, $H(X|Y)$;
- et de façon symétrique, l'information reçue, diminuée de l'incertitude sur le symbole reçu y quand le symbole émis x est connu, $H(Y|X)$.

5 Description mathématique d'une communication

Un canal de transmission peut être vu comme un système qui reçoit en entrée des symboles émis par une source d'alphabet $\Omega_X = \{x_i\}_{i=1}^n$ et qui donne en sortie des symboles de l'alphabet du récepteur $\Omega_Y = \{x_i\}_{i=1}^m$ (éventuellement différent de Ω_X). Lors de la transmission des erreurs peuvent se produire de façon aléatoire. Ainsi, le lien entre un symbole émis et un symbole reçu est incertain. On peut alors parler de **canal avec bruit.**

Dans ce qui suit, nous ne nous intéressons pas à la nature de ce bruit ni à ses causes possibles. *La théorie de l'information s'intéresse aux conséquences de l'incertitude introduite par ce bruit sur l'exactitude de l'information reçue à la sortie d'un canal de transmission.*

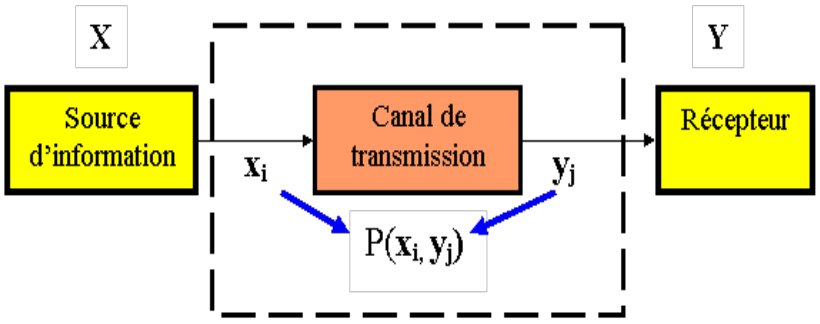

FIGURE 5.1 – Canal de transmission

Quelles sont les mesures de quantité d'information qui peuvent être associées à l'ensemble "source-canal-récepteur" ? Commençons par un exemple simple.

Soient une source binaire X d'alphabet $\Omega = \{0, 1\}$ et de distribution de probabilité uniforme : $P(1) = P(0) = 0.5$. Soit un récepteur Y de même alphabet. Imaginons, qu'à chaque symbole émis, le canal de transmission a la probabilité de $\frac{1}{4}$ de faire une erreur. On peut représenter alors le fonctionnement de ce canal par le schéma suivant :

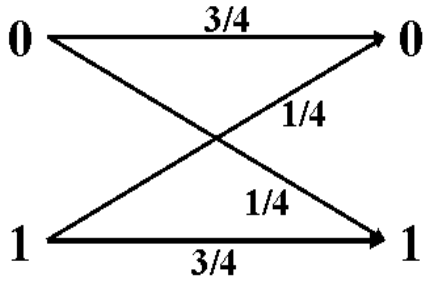

FIGURE 5.2 – Canal de transmission

Soient X et Y les variables aléatoires associées respectivement à la source et au récepteur. La probabilité de l'erreur donnée permet d'établir les probabilités conditionnelles suivantes :

$$P(Y=0|X=0)=\frac{3}{4} \quad P(Y=1|X=0)=\frac{1}{4}$$

$$P(Y=0|X=1)=\frac{1}{4} \quad P(Y=1|X=1)=\frac{3}{4}$$

En connaissant également la distribution de probabilité de la source X on peut en déduire la distribution conjointe. En effet,

$$P_{XY}(0,0)=P(X=0)P(Y=0|X0)=\frac{3}{4}\cdot\frac{1}{2}=\frac{3}{8} \quad P_{XY}(0,1)=P(X=0)P(Y=1|X=0)=\frac{1}{8}$$

$$P_{XY}(1,0)=P(X=1)P(Y=0|X=1)=\frac{1}{8} \qquad P_{XY}(1,1)=P(X=1)P(Y=1|X=1)=\frac{3}{8}$$

Enfin, on peut calculer les distributions marginales de X et Y :

$$P_X(0)=P_X(1)=\frac{1}{2}, \quad P_Y(0)=P_Y(1)=\frac{3}{8}+\frac{1}{8}=\frac{1}{2}$$

À partir de ces différentes distributions des probabilités, on peut maintenant calculer les entropies $H(X),\ H(Y),\ H(X,Y),\ H(X|Y)$ et enfin l'information mutuelle $I(X;Y)$:

$$H(X)=H(Y)=1$$

L'entropie de chacune des variables X et Y décrit la difficulté moyenne de prédire le symbole émis (ou reçu).

$$H(X,Y)=-\left(2\frac{1}{8}\log_2\left(\frac{1}{8}\right)+2\frac{3}{8}\log_2\left(\frac{3}{8}\right)\right)\simeq 1.8$$

L'entropie mutuelle décrit l'information moyenne apportée par l'observation d'un couple de symboles x, le symbole émis et y, le symbole reçu.

$$H(Y|X)=-\left(2\frac{1}{8}\log_2\left(\frac{1}{4}\right)+2\frac{3}{8}\log_2\left(\frac{3}{4}\right)\right)\simeq 0.8$$

L'entropie conditionnelle exprime la difficulté moyenne de prévoir le symbole reçu lorsqu'on connaît celui qui a été émis.

$$I(X;Y) = H(X) - H(Y|X) = 1 - 0.8 = 0.2$$

6 Canal discret stationnaire, sans mémoire

Supposons que l'alphabet de la source, en entrée est composé de n symboles $\Omega_X = \{x_i,\ i = 1, \ldots, n\}$ et que l'alphabet du récepteur, en sortie, est composé de m symboles $\Omega_Y = \{y_j,\ j = 1, \ldots, m\}$.

Dans la suite de ce cours, nous allons considérer seulement un cas particulier de canaux de communication, **les canaux discrets sans mémoire et stationnaires**. Un tel canal est décrit par la donnée de la matrice de probabilités conditionnelles, appelée **matrice de transition**

$$p_{i,j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme **"sans mémoire"** signifie que la réception à tout instant d'un symbole Y ne dépend que du symbole émis X. En particulier, chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme **"stationnaire"** signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps. Ainsi, à tout instant de transmission, cette matrice est la même.

En pratique, pour décrire les caractéristiques du bruit d'un canal on dispose souvent de la matrice de transition $P_{Y|X}$: $n \times m$, c'est à dire de la distribution conditionnelle de la sortie sachant l'entrée. Dans ces cas là on connaît également la distribution de probabilité de la source $P_X = (P(x_i))_{i=1}^n$. En connaissant ces deux distributions on peut déduire toutes les autres distributions associées au couple (X, Y) de l'émetteur et récepteur :

- Distribution de probabilité conjointe :

$$\forall (i,j),\quad P(x_i, y_j) = P(x_i)P(y_j|x_i)$$

- Distribution marginale de Y :

$$\forall j,\quad P(y_j) = \sum_{i=1}^{n} P(y_j|x_i)P(x_i)$$

- Distribution conditionnelle $P[X|Y]$:

$$\forall (i,j),\quad P(x_i|y_j) = P(x_i, y_j)/P(y_j)$$

Lorsque toutes ces distributions sont connues, on peut associer à un système de communication "source - canal - récepteur" différentes entropies :

$H(X)$**. L'entropie de la source** Elle représente l'information moyenne par symbole de la source ou encore la difficulté moyenne de prédire le symbole émis.

$H(Y)$**. L'entropie du récepteur** Elle représente l'information moyenne par symbole reçu ou encore la difficulté moyenne de prédire le symbole reçu.

$H(X,Y)$**. L'entropie conjointe "source-récepteur"** Elle représente l'incertitude moyenne du système de communication dans son ensemble ou encore la quantité de l'information moyenne par paire "symbole émis - symbole reçu".

$H(X|Y)$**. L'entropie conditionnelle de la source, sachant le symbole reçu** Elle représente l'incertitude moyenne sur le symbole émis lorsqu'on connaît le symbole reçu.

$H(Y|X)$**. L'entropie conditionnelle du récepteur, sachant le symbole émis** Elle représente l'incertitude moyenne sur le symbole reçu lorsqu'on connaît le symbole émis.

$I(X;Y)$**. L'information mutuelle moyenne** Elle représente la quantité moyenne d'information par symbole transmise à travers le canal.

6.1 Exemple complet

Soient une source d'un alphabet de 5 symboles $\Omega_X = \{x_1, x_2, x_3, x_4, x_5\}$ et un récepteur d'alphabet ayant quatre symboles $\Omega_Y = \{y_1, y_2, y_3, y_4\}$. Supposons que la matrice de probabilités conjointes associée à un canal est connue :

$P(X,Y) =$

	y_1	y_2	y_3	y_4
x_1	0.25	0	0	0
x_2	0.1	0.3	0	0
x_3	0	0.05	0.10	0
x_4	0	0	0.05	0.10
x_5	0	0	0.05	0

Calculons toutes les autres distributions de probabilités associées :

Distribution marginale de la source On utilise la définition

$$\forall i = 1, \ldots, 5,\ p(x_i) = \sum_{j=1}^{4} p(x_i, y_j)$$

Ainsi, en faisant les sommes des éléments de chaque **ligne** de la matrice $P(X,Y)$ on trouve :

$$\begin{array}{ll} p(x_1) = 0.25 & p(x_2) = 0.1 + 0.3 = 0.4 \\ p(x_3) = 0.05 + 0.10 = 0.15 & p(x_4) = 0.05 + 0.10 = 0.15 \\ p(x_5) = 0.05 & \end{array}$$

Distribution marginale du récepteur On utilise la définition

$$\forall j = 1, \ldots, 4,\ p(y_j) = \sum_{i=1}^{5} p(x_i, y_j)$$

Ainsi, en faisant les sommes des éléments de chaque **colonne** de la matrice $P(X,Y)$ on trouve :

$$\begin{array}{ll} p(y_1) = 0.25 + 0.10 = 0.35 & p(y_2) = 0.3 + 0.05 = 0.35 \\ p(y_3) = 0.10 + 0.05 + 0.05 = 0.2 & p(y_4) = 0.10 \end{array}$$

Distribution conditionnelle $P(X|Y)$ On utilise la définition

$$\forall i = 1, \ldots, 5,\ \forall j = 1, \ldots, 4,\ p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)}$$

On trouve la matrice

$$P(X|Y) = \begin{pmatrix} 5/7 & 0 & 0 & 0 \\ 2/7 & 6/7 & 0 & 0 \\ 0 & 1/7 & 1/2 & 0 \\ 0 & 0 & 1/4 & 1 \\ 0 & 0 & 1/4 & 0 \end{pmatrix}$$

Distribution conditionnelle $P(Y|X)$ On utilise la définition

$$\forall i = 1, \ldots, 5,\ \forall j = 1, \ldots, 4,\ p(y_j|x_i) = \frac{p(x_i, y_j)}{p(x_i)}$$

On trouve la matrice

$$P(Y|X) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 \\ 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 1/3 & 2/3 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Calculons maintenant les entropies.

$H(X)$**. L'entropie de la source**

$$\begin{aligned} H(X) &= -\sum_{i=1}^{5} p_i \log_2 p_i = \\ &= -0.25 \log_2 0.25 - 0.4 \log_2 0.4 - 2 \cdot 0.15 \log_2 0.15 - 0.05 \log_2 0.05 \simeq 2.066 \end{aligned}$$

$H(Y)$**. L'entropie du récepteur**

$$\begin{aligned} H(Y) &= -\sum_{i=1}^{4} p_i \log_2 p(y_i) = \\ &= -2 \cdot 0.35 \log_2 0.35 - \cdot 0.20 \log_2 0.20 \\ &- 0.10 \log_2 0.10 \simeq 1.856 \end{aligned} \tag{5.7}$$

$H(X, Y)$**. L'entropie conjointe "source-récepteur"**

$$\begin{aligned} H(X, Y) &= -\sum_{i=1}^{5} \sum_{j=1}^{4} p(x_i, y_j) \log_2 p(x_i, y_j) = \\ &= -0.25 \log_2 0.25 - 3 \cdot 0.1 \log_2 0.1 \\ &- 3 \cdot 0.05 \log_2 0.05 - 0.3 \log_2 0.3 \simeq 2.665 \end{aligned}$$

$H(X|Y)$. L'entropie conditionnelle de la source, sachant le symbole reçu

$$\begin{aligned} H(X|Y) &= -\sum_{i=1}^{5}\sum_{j=1}^{4} p(x_i,y_j)\log_2\frac{p(x_i,y_j)}{p(y_j)} = \\ &= -0.25\log_2\frac{5}{7} - 0.1\log_2\frac{2}{7} - 0.1\log_2\frac{1}{2} \\ &- 0.05\left(\log_2\frac{1}{7}+\log_2\frac{1}{4}+\log_2\frac{1}{4}\right) - 0.3\log_2\frac{6}{7} \simeq 0.809 \end{aligned} \tag{5.8}$$

$H(Y|X)$. L'entropie conditionnelle du récepteur, sachant le symbole émis

$$\begin{aligned} H(Y|X) &= -\sum_{i=1}^{5}\sum_{j=1}^{4} p(x_i,y_j)\log_2\tfrac{p(x_i,y_j)}{p(x_i)} = \\ &= -0.1(\log_2\tfrac{1}{4}+2\log_2\tfrac{2}{3}) - 2\cdot 0.05\log_2\tfrac{1}{3} - 0.3\log_2\tfrac{3}{4} = 0.6 \end{aligned}$$

6.2 Exemples fréquents de canaux

Il existe quelques classes de canaux qu'il est plus facile à analyser. Nous en donnons ici quelques exemples (tirés de [4] et [2]).

6.2.1 Canal avec entrée et sortie indépendantes

Il s'agit d'un canal d'alphabets d'entrée et de sortie respectivement $\Omega_X = \{x_i,\ i = 1,\ldots,n\}$ et $\Omega_Y = \{y_j,\ j = 1,\ldots,m\}$ tel que, quel que soit le symbole émis, on peut recevoir n'importe quel symbole y_j avec équiprobabilité :

$$p(y_j|x_i) = p(y_j) = \frac{1}{m}$$

Ainsi ma matrice de transition est de la forme :

$$P[Y|X] = \begin{pmatrix} 1/m & 1/m & \cdots & 1/m \\ 1/m & 1/m & \cdots & 1/m \\ \vdots & \cdots & \ddots & 1/m \\ 1/m & \cdots & 1/m & 1/m \end{pmatrix}$$

Étant donnée la distribution de probabilité de la source : $P(x_i) = p_i,\ i = 1,\ldots,n\ \sum_{i=1}^{n} p_i = 1$ on en déduit les probabilités conjointes :

$$p(x_i,y_j) = p(x_i)p(y_j) = \frac{p_i}{m} = q_i,\quad i = 1,\ldots,n,\quad j = 1,\ldots,m$$

La matrice de probabilités conjointes a alors m colonnes identiques

$$P[X,Y] = \begin{pmatrix} q_1 & q_1 & \cdots & q_1 \\ q_2 & q_2 & \cdots & q_2 \\ \vdots & \vdots & \ddots & \vdots \\ q_n & q_n & \cdots & q_n \end{pmatrix}$$

La distribution conditionnelle de la source sachant le récepteur se calcule comme suit :

$$p(x_i|y_j) = p(x_i) = m \cdot q_i$$

Pour les entropies on a

$$\begin{aligned} H(X,Y) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j) log p(x_i,y_j) = -m\sum_{i=1}^{n} q_i \log_2 q_i \\ H(X) &= -\sum_{i=1}^{n} p_i \log_2 p_i \\ H(Y) &= \log_2 m \\ H(X|Y) &= H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \\ H(Y|X) &= H(Y) = \log_2 m \\ I(X,Y) &= H(X) - H(X|Y) = 0 \end{aligned}$$

L'interprétation de ces formules est la suivante : un tel canal, ne transporte aucune information entre la source et le récepteur ($I(X;Y) = 0$). Si un canal sans bruit peut être considéré comme le cas idéal d'un canal sans pertes alors un canal à entrée et sortie indépendantes comme ayant la perte maximale d'informations.

6.2.2 Canal sans pertes

Soit, comme ci-haut, X la variable aléatoire associée à la source et Y la variable aléatoire associée au message reçu. On dit que le canal est sans pertes si $H(X|Y) = 0$. Autrement dit, le symbole envoyé est identifié sans aucune ambiguïté par le symbole reçu. Si l'alphabet de la source est $\Omega_X = \{x_i,\ i = 1,\ldots,M\}$ et si celui du récepteur est $\Omega_Y = \{y_j,\quad j = 1,\ldots,N\}$ avec $N > M$ alors dans un canal sans pertes il est possible de partitionner l'alphabet Ω_Y en M sous-ensembles disjoints $B_i,\quad i = 1,\ldots,M$ tels que

$$P[x = x_i | y \in B_i] = 1$$

Dans ce cas, nous avons

$$I(X|Y) = H(X) - H(X|Y) = H(X)$$

6.2.3 Canal déterministe

On dit que le canal est sans pertes si $H(Y|X) = 0$. Autrement dit, le symbole reçu est déterminé de façon unique par le symbole envoyé. Si l'alphabet de la source est $\Omega_X = \{x_i,\ i = 1, \ldots, M\}$ et si celui du récepteur est $\Omega_Y = \{y_j,\ \ j = 1, \ldots, N\}$ on a $P(y_j|x_i)$ est égal à 0 ou 1. Dans ce cas, nous avons

$$I(X|Y) = I(Y|X) = H(Y) - H(Y|X) = H(Y)$$

6.2.4 Canal sans bruit

L'absence de bruit dans un canal signifie que la transmission est exacte, sans erreurs. un canal sans bruit est donc un canal sans pertes et déterministe. Le canal peut alors être vu comme une mise en correspondance bi-univoque (bijective) des deux alphabets. Cela se traduit par une matrice de transition identité :

$$P[Y|X] = I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

La matrice de probabilités conjointes est alors diagonale

$$P[X,Y] = I_n = \begin{pmatrix} p(x_1,y_1) & 0 & \cdots & 0 \\ 0 & p(x_2,y_2) & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & p(x_n,y_n) \end{pmatrix}$$

Il est facile de vérifier que les entropies sont :

$$H(X,Y) = H(X) = H(Y) = -\sum_{i=1}^{n} p(x_i,y_i) \log_2 p(x_i,y_i)$$

et que

$$H(X|Y) = H(Y|X) = 0$$

Enfin,

$$I(X;Y) = H(X) - H(X|Y) = H(X)$$

On peut facilement interpréter ces relations. En effet, dans un canal sans bruit à chaque symbole émis correspond un et une seul symbole reçu. Alors lorsque l'on connaît le symbole émis, on connaît **sans aucune incertitude** le symbole reçu et *vice versa*. Ainsi, les entropies conditionnelles moyennes sont nulles car elle mesurent précisément l'incertitude moyenne sur le symbole émis (resp. reçu) sachant le symbole reçu (resp. émis). C'est aussi pour cette raison que les incertitudes moyennes par symbole à la source, $H(X)$, et à la réception, $H(Y)$, sont les mêmes.

6.2.5 Canal inutile

Un canal est inutile si $I(X|Y) = 0$. Autrement dit, le canal ne transporte aucune information. Cette condition est équivalente à $H(X) = H(X|Y)$: la connaissance de symbole reçu ne modifie pas (n'apporte aucune information supplémentaire) l'incertitude sur le symbole émis.

6.2.6 Canal symétrique

Un canal symétrique est caractérisé par une propriété particulière de sa matrice de transition $P(Y|X)$. Toutes les lignes de cette matrice sont équivalentes entre elles par permutation et toutes les colonnes également. Cela veut dire que toutes les lignes sont obtenues par permutation d'une seule d'entre elles, et toutes les colonnes sont obtenues par permutations d'une seule d'entre elles. Voici un exemple d'une telle matrice

$$P(Y|X) = \begin{array}{c|ccc} & y_1 & y_2 & y_3 \\ \hline x_1 & 1/2 & 1/6 & 1/3 \\ x_2 & 1/6 & 1/3 & 1/2 \\ x_3 & 1/3 & 1/2 & 1/6 \end{array}$$

Proposition 5.4. *Dans un canal symétrique l'entropie conditionnelle $H(Y|X)$ ne dépend pas de la distribution des probabilités de X.*

Preuve Proposition 5.4

Soit $x_i \in \Omega_X$ un symbole quelconque de l'alphabet de la source. Calculons

$$H(Y|X = x_i) = -\sum_{j=1}^{M} p(y_j|x_i) \cdot \log_2(p(y_j|x_i))$$

Or pour un canal symétrique toutes les lignes de la matrice de transition sont obtenues par permutation à partir d'une seule d'entre elles. Ainsi, toutes les lignes de la matrices contiennent exactement le même ensemble de valeurs. Notons les $\alpha_1, \ldots, \alpha_M$. Nous avons alors pour tout $x_i \in \Omega_X$

$$H(Y|X = x_i) = -\sum_{j=1}^{M} \alpha_j \cdot \log_2(\alpha_j)$$

Ainsi la valeur de $H(Y|x_i)$ ne dépend pas de x_i. Alors on a

$$H(Y|X) = \sum_{i=1}^{N} p(x_i) H(Y|x_i) = -\sum_{j=1}^{M} \alpha_j \cdot \log_2(\alpha_j) \sum_{i=1}^{N} p(x_i) = -\sum_{j=1}^{M} \alpha_j \cdot \log_2(\alpha_j)$$

C.Q.F.D

Dans cette classe de canaux le plus connu est le canal symétrique binaire. Les deux alphabets, de source et de récepteur sont identiques et coposés de deux caractères : $\Omega_X = \Omega_Y = \{0, 1\}$. La matrice de transition ets alors égale à

$$P(Y|X) = \begin{pmatrix} 1-\alpha & \alpha \\ \alpha & 1-\alpha \end{pmatrix}$$

où $\alpha \in [0, 1]$ est la probabilité d'erreur de transmission.

7 Capacité d'un canal

Nous avons donné plus haut la définition de l'information mutuelle moyenne $I(X;Y)$ comme mesure de l'information transmise en moyenne par symbole à travers le canal. On peut remarquer que cette quantité dépend du canal mais aussi de la source d'information. On introduit alors une nouvelle mesure, qui ne décrit que le canal : la capacité de canal.

Définition 5.4 (Capacité d'un canal)**.** La capacité d'un canal est définie par

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} (H(X) - H(X|Y))$$

Le maximum est pris sur toutes les distributions de probabilité possibles de la source.

Exemple 5.3 (Capacité d'un canal sans bruit)**.** Considérons un canal sans bruit et une source d'alphabet $\Omega = \{x_i,\ i = 1, \ldots, n\}$. Nous avons déjà vu (voir 6.2.4) que dans ce cas $I(X;Y) = H(X)$. Alors

$$C = \max_{P(X)} I(X;Y) = \max H(X)$$

Or nous avons également vu que l'entropie d'une source est maximale lorsque tous les symboles de l'alphabet sont équiprobables. Sa valeur maximale est alors égale à $\log_2 n$. Ainsi on en déduit que pour un canal sans bruit d'alphabet de n caractères

$$C = \log_2 n$$

7.1 Capacité d'un canal symétrique

Si dans le cas général le calcul de capacité peut s'avérer difficile, il est possible d'obtenir des expressions explicites dans les cas de quelques canaux particuliers que nous avons étudiés précédemment. Ici nous allons développer le cas très utile d'un canal symétrique.

Proposition 5.5. *Soit un couple émetteur-récepteur (X, Y) avec les alphabets $\Omega_X = \{x_i,\ i = 1, \ldots, n\}$ et $\Omega_X = \{x_i,\ i = 1, \ldots, m\}$. Soit un canal symétrique de matrice de transition donnée*

$$P(Y|X) = \begin{pmatrix} p(y_1|x_1) & p(y_2|x_1) & \cdots & p(y_{m-1}|x_1) & p(y_m|x_1) \\ p(y_1|x_2) & \cdots & \cdots & \cdots & p(y_m|x_2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ p(y_1|x_n) & \cdots & \cdots & \cdots & p(y_m|x_n) \end{pmatrix}$$

Alors la capacité de ce canal est

$$C = \log_2 m + \sum_{j=1}^{m} p(y_j|x_1)$$

Preuve la Proposition 5.5

Nous allons utiliser la proposition 5.4 qui établit une propriété remarquable de l'entropie conditionnelle $H(Y|X)$ dans un canal symétrique : on sait que $H(Y|X)$ ne dépend pas de la distribution de X.

Alors si l'on cherche à maximiser l'information transmise par le canal

$$I(Y, X) = H(Y) - H(Y|X)$$

nous devons trouver la distribution de X qui maximise $H(Y)$ seulement puisque le deuxième terme de cette expression de dépend par de la distribution de X.

Nous savons, d'après les propriétés générales de l'entropie qu'elle atteint sa valeur maximale quand la variable aléatoire correspondante est distribuée uniformément. Montrons que si la source, X, est distribuée uniformément, alors la variable du récepteur, Y est aussi distribuée uniformément et donc $H(Y)$ est maximale.

Supposons donc que $p(x_i) = \dfrac{1}{n}, \quad \forall x_i \in \Omega_X$. Alors, en utilisant la règle de Bays, on trouve :

$$p(y_j) = \sum_{i=1}^{n} p(x_i, y_j) = \sum_{i=1}^{n} p(x_i) p(y_j|x_i) = \frac{1}{n} \sum_{i=1}^{n} p(y_j|x_i)$$

Or, dans la matrice de transition d'un canal symétrique toutes les colonnes sont obtenues par permutation de l'une d'entre elles. Donc la somme

$$\sum_{i=1}^{n} p(y_j|x_i) = \sum_{i=1}^{n} p(y_1|x_i)$$

ne dépend pas de y_j. Ainsi on a montré que si la distribution de X est uniforme toutes les probabilités $p(y_j)$ sont égales entre elles. Donc dans ce cas $H(Y)$ est maximale et égale à $\log_2 m$.

La formule de la proposition suit alors immédiatement de ce fait et de la proposition 5.4 dans laquelle on a établit la formule pour $H(Y|X)$. **C.Q.F.D**

Chapitre 6

Codage de canal en présence de bruit. Codes correcteurs d'erreurs.

1 Second théorème de Shannon

1.1 Codage de canal

Dans cette partie nous allons poser et étudier le problème de codage dans le contexte de transmission via un canal avec bruit. La différence par rapport au codage sans bruit est dans le fait que lors de la transmission des messages des erreurs peuvent se produire de façon aléatoire. Ces erreurs sont caractérisées par la donnée de la matrice de transition du canal.

Notre problème sera alors d'évaluer la probabilité d'erreur de transmission. Nous allons montrer que cette probabilité dépend non seulement des caractéristiques probabilistes du canal et de la source mais aussi du choix du codage et de décodage.

Alors le problème de choix de meilleur code en présence de bruit sera reformulé pour prendre en compte non seulement la vitesse de transmission moyenne mais aussi la probabilité d'erreur.

1.1.1 Règle de décodage d'un canal avec bruit.

En absence de bruit de transmission la fonction de décodage était naturellement définie comme la transformation inverse de la fonction de codage. C'était par définition l'unique façon de retrouver exactement le message émis.

En présence de bruit il existe une incertitude sur le message émis lorsqu'on observe le message reçu. Autrement dit, un message reçu peut correspondre à plusieurs messages en entrée avec une distribution de probabilité qui peut être déduite à partir de la matrice de transition. Pour traiter ce problème et quantifier la probabilité d'erreur de transmission nous allons introduire la notion de règle de décodage de canal.

Définition 6.1 (Règle de décodage)**.** Soit un canal avec un alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$ et un alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$. La **règle de décodage** de canal est une fonction déterministe

$$g : \{y_1, \ldots, y_D\} \to \{x_1, \ldots, x_M\}$$

qui à chaque symbole reçu y_j associe un symbole de l'alphabet d'entrée $x_j^* = g(y_j)$.

On peut interpréter cela sous forme de règle de décision : si le symbole y_j est reçu il est décodé comme x_j^*. Soient les variables aléatoires X, Y et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé. Considérons un exemple. Soit l'alphabet d'entrée de taille 3 et celui de sortie de taille 3. Supposons que l'alphabet d'entrée a la distribution de probabilité suivante

$$p(x_1) = 1/2,\ p(x_2) = p(x_3) = 1/4$$

Les probabilités de transition et la règle de décodage sont illustrées par le schéma sur la figure 6.1.

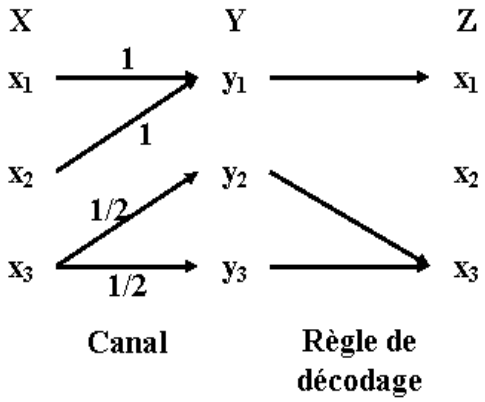

FIGURE 6.1 – Un exemple de règle de décision

Considérons la règle de décision g définie par

$$g(y_1) = x_1,\ g(y_1) = x_2,\ \ g(y_3) = x_2$$

Le seul cas où une erreur a lieu est l'émission de symbole x_2. Ainsi la probabilité d'erreur est égale à la probabilité d'émission de x_2 donc à 1/4.

Dans le cas général soit E l'événement correspondant à un erreur lors de transmission d'un seul symbole. Comment peut on calculer $P[E]$? Soient les variables aléatoires X, Y et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.

Tout d'abord, on peut décomposer cette probabilité selon la formule de Bays :

$$P(E) = \sum_{i=1}^{n} p(x_i)p(E|x_i)$$

Sachant que le symbole x_i est transmis, l'erreur se produit lorsque la fonction fonction de décodage renvoie un caractère différent de x_i. Ainsi dans ce cas l'événement E équivaut à $g(Y) \neq x_i$. On a donc $p(E|x_i) = P(g(Y) \neq x_i|x_i)$. Cette dernière probabilité se décompose en somme selon les différentes valeurs de Y :

$$p(E|x_i) = P(g(Y) \neq x_i|x_i) = \sum_{j=1}^{d} p(Y = y_j \text{ et } g(y_j) \neq x_i|x_i)$$

Etant donné que la règle de décodage est déterministe on a

$$p(Y = y_j \text{ et } g(y_j) \neq x_i|x_i) = \begin{cases} p(y_j|x_j), & \text{si } g(y_j) \neq x_i \\ 0 & \text{si } g(y_j) = x_i \end{cases} = p(y_j|x_j)(1 - \delta_{g(y_j), x_i})$$

Ici $\delta_{ik} = \begin{cases} 1, & \text{si } i \neq k \\ 0 & \text{si } i = k \end{cases}$ est le symbole de Kronecker.

Nous obtenons ainsi la probabilité d'erreur conditionnelle pour la transmission d'un seul symbole :

$$p(E|x_i) = \sum_{j=1}^{d} p(y_j|x_j)(1 - \delta_{g(y_j), x_i}).$$

1.1.2 Notion de code de canal.

On peut généraliser les raisonnements de la section précédente au cas où il s'agit de transmettre des messages par blocks de longueur donné l.

Dans la suite un canal est modélisé par le triplet $X, Y, P(Y|X)$ où X est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$ et Y est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$. $P(Y|X)$ est la matrice de transition du canal.

On peut associer à La transmission d'un mot de longueur l une variable aléatoire $X^{(l)}$ à valeurs dans Ω_X^l. Cette variable peut être interprétée comme observation simultanée de l variables aléatoires indépendantes et toutes distribuées comme X :

$$X^{(l)} = (X_1, \ldots, X_l)$$

De même, on peut associer à la réception d'un mot de l caractères la variable aléatoire $Y^{(l)}$ à valeurs dans Ω_Y^l. Cette variable peut être interprétée comme observation simultanée de l variables aléatoires indépendantes et toutes distribuées comme Y :

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

On peut considérer alors la transmission d'un message de longueur l comme un nouveau canal $X^{(l)}, Y^{(l)}, P(X^{(l)}|X^{(l)})$, appelé **$l$ème extension de canal** initial. la matrice de transition de ce nouveau canal peut être déduite de

$$P((y_1, \ldots, y_l)|(x_1, \ldots, x_l)) = \prod_{k=1}^{l} p(y_k|x_k)$$

Définition 6.2 (Code de canal)**.** Soit un canal $X, Y, P(Y|X)$. Un code (n, l) pour ce canal est un couple (W, g) où

1. $W = \{w_1, \ldots, w_n\}$ est un ensemble de mots de longueur l dans l'alphabet d'entrée Ω_X, appelés mots du code.
2. g est une règle de décodage
$$g : (\Omega_Y)^l \to W$$
qui associe à toute séquence reçue de longueur l dans l'alphabet de sortie Ω_Y un des mots du code.

Définition 6.3 (Probabilité d'erreur conditionnelle)**.** Soient un canal $X, Y, P(Y|X)$ et un code (n, l) pour ce canal (W, g). Pour chaque mot du code w_i on définit la probabilité d'erreur conditionnelle
$$\lambda_i^{(l)} = P[E|w_i] = \sum_{(y_1,\ldots,y_l)\in(\Omega_Y)^l} P((y_1, \ldots, y_l)|w_i)(1 - \delta_{g(y_1,\ldots,y_l),w_i}).$$

Définition 6.4 (Probabilité d'erreur moyenne)**.** Soient un canal $X, Y, P(Y|X)$ et un code (n, l) pour ce canal (W, g). On définit la probabilité d'erreur moyenne (algébrique) du code par :
$$\lambda^{(l)} = \frac{1}{n}\sum_{i=1}^{n} \lambda_i^{(l)}.$$

Définition 6.5 (Probabilité d'erreur maximale)**.** Soient un canal $X, Y, P(Y|X)$ et un code (n, l) pour ce canal (W, g). On définit la probabilité d'erreur maximale du code par :
$$\lambda^{(l)} = \max_{i=1,\ldots,n} \lambda_i^{(l)}.$$

Définition 6.6 (Débit de communication d'un code)**.** Soient un canal $X, Y, P(Y|X)$ et un code (n, l) pour ce canal (W, g). On définit le débit de communication du code par :
$$R = \frac{log(n)}{l}.$$
l'unité de mesure est Shannon par symbole transmis.

1.2 Second théorème de Shannon

Nous avons maintenant pourvoir poser le problème de codage en présence de bruit :

> Étant donné un canal $X, Y, P(Y|X)$ de capacité C est il possible de transmettre des messages avec un débit aussi proche que possible de C et avec une probabilité d'erreur aussi petite que possible ?

La réponse à ce problème est donnée par le théorème suivant :

Théorème 6.1 (Second théorème de Shannon)**.** *Soit un canal $X, Y, P(Y|X)$ de capacité $C > 0$. Pour tout $R < C$ il est possible de trouver un code de canal avec un débit R et une probabilité d'erreur aussi petite que possible. Plus précisément, il existe une suite de codes $(M(l), l)$ tels que $M(l) = 2^{lR}$ telle que*

$$\lim_{l \to \infty} \lambda^{(l)} = 0$$

2 Codes correcteurs d'erreurs

Le second théorème de Shannon établit l'exitance de codes de canal permettant de transmettre à un débit aussi proche qu'on souhaite de la capacité du canal et une probabilité d'erreur arbitrairement petite. mais, comme dans le cas de codage de source, le théorème ne donne aucune indication sur la façon de construire de tels codes. En pratique, il existe de nombreuses familles de codes permettant d'approcher au mieux les performances optimales, sans pour autant alourdir les temps de codage et décodage.

Dans cette section nous allons présenter une classe particulière de codes de canal qui permet de détecter et corriger assez bien des erreurs de transmission. Il s'agit de codes binaires linéaires. Nous introduirons d'abord le groupe G_k formé de k-uplets de bits. Ensuite, on introduira la distance de Hamming qui permet d'évaluer efficacement les erreurs de transmission. Enfin, après quelques définitions générales liées à la détection et correction d'erreurs nous allons introduire les codes linéaires et leurs propriétés.

2.1 Généralités

Dans la suite de la section nous allons étudier les codes par blocks binaires. On suppose qu'on dispose d'un message à transmettre sous forme d'une suite binaire de longueur M. Pour le codage le message sera découpé en blocks de k bits. On suppose dans la suite que tous les blocks de k bits possibles sont équiprobables. Le codage consiste à remplacer chaque block de k bits par un mot binaire de longueur $n \geq k$. Ainsi on définira plus tard un code comme une application

qui associe à chaque block de longueur k un mot binaire de longueur n. On l'appellera table de code.

Après une transmission à travers un canal bruité le mot reçu peut être erroné et ne correspondre à aucun des mots de la table de code. Nous allons donc décrire ci-dessous une règle de décodage au sens Cette règle utilise la notion de distance de Hamming que nous allons d'abord expliciter.

L'objectif principal de conception des codes se pose alors en termes de capacité de détection et de correction des erreurs de transmission. Nous allons insister sur le fait que cette capacité est une propriété intrinsèque d'un code et non le fait d'un choix particulier de règle de décodage.

2.2 Un premier exemple illustratif

Nous allons étudier ici une procédure très simple de codage et décodage permettant de détecter et même corriger des erreurs de transmission : le code à répétition.

Codage. Le principe est très simple : chaque bit du message à transmettre sera répété n fois. Ainsi, les blocks ici sont de taille $k = 1$ et les mots de code seront de taille n. Prenons l'exemple de $n = 3$. La table de code est alors :

s	w
0	000
1	111

Canal de transmission. Supposons que nous utilisons pour la transmission un canal binaire symétrique dont la probabilité d'erreur est $0 < \alpha < 0.5$. La matrice de transmission est alors

$$P(Y|X) = \begin{pmatrix} 1-\alpha & \alpha \\ \alpha & 1-\alpha \end{pmatrix}$$

Chaque bit transmis a la probabilité α d'être reçu comme son contraire.

Règle de décodage. A la réception, le message sera analysé par blocks de 3 bits. Remarquons ici que dans notre code il n'y a que deux mots : 000 et 111. Cependant, compte tenu des erreurs de transmission l'ensemble de mots possibles à la réception contient tous les 8 mots de 3 bits. Si on reçoit l'un des mots du code le décodeur le remplacera pas le symbole correspondant. Mais que doit on faire si on reçoit 010 ? Il est évident dans ce cas que des erreurs de transmission se sont produites. La règle de décodage doit donc permettre de décider à quel mot de code (à quel symbole) correspond cette séquence. Une règle simple à mettre en oeuvre dans cette situation est la règle du vote majoritaire :

- Si la majorité des 3 bits reçus sont 0 on considère qu'il s'agit du mot de code 000 et on décode donc par 0.
- Si la majorité des bits sont 1 on considère que le mots de code transmis est 111 est on remplace par 1.

Le nombre de bits étant impaire, cette règle est non ambiguë. Voici une exemple d'application de ce schéma sur un message de 5 bits.

Message original	1	0	1	0	1
Message codé	111	000	111	000	111
Message reçu	101	101	110	000	011
Message décodé	1	1	1	0	1

Cet exemple montre qu'il est possible de détecter et corriger les erreurs de transmission presque toujours. En effet, si un mot subit deux erreurs de transmission ou plus le décodage sera inexact.

Revenons à la signification de la règle du vote majoritaire. Dans notre code il n'y a que deux mots 000 et 111. Pour un mot reçu de trois bits, le vote majoritaire correspond à lui associer celui des deux mots de code qui a le plus de bits en commun avec ce mot. Cette règle peut donc être étendue à d'autres types de codes, en prenant en compte les ressemblances, bit par bit entre deux mots. Ce sera fait plus loin à l'aide de la notion de distance de Hamming.

Donnons maintenant une interprétation probabiliste de la règle du vote majoritaire. Considérons un mot reçu $t = t_1t_2t_3$ de trois bits. Evaluons les probabilités conditionnelles $P(t|s = 0)$ et $P(t|s = 1)$ pour essayer de savoir laquelle des deux est plus importante.

$$P(t = t_1t_3t_3|s = 0) = P(t = t_1t_2t_3|w = 000) = P(t_1|0)P(t_2|0)P(t_3|0)$$

car nous supposons que les symboles sont transmis de façon indépendante les uns des autres. D'après la matrice de transmission de notre canal

$$P(t_i|0) = \begin{cases} 1-\alpha, & \text{si } t_1 = 0 \quad \text{transmission correcte} \\ \alpha, & \text{si } t_1 = 1 \quad \text{transmission erronée} \end{cases}, \quad i = 1, 2, 3$$

Soit l le nombre de 0 dans t. Alors il y a l bits correctes et $3 - l$ bits à 1 donc transmis avec erreur. Alors

$$P(t = t_1t_3t_3|s = 0) = (1-\alpha)^l\alpha^{3-l}$$

Pour la même séquence $t = t_1t_2t_3$ on a alors

$$P(t = t_1t_3t_3|s = 1) = (\alpha)^l(1-\alpha)^{3-l}$$

car si c'est 1 qui est transmis alors chaque 0 dans t correspond à une erreur, donc à la probabilité α. Pour comparer les deux probabilités, considérons leur rapport

$$\frac{P(t = t_1t_3t_3|s = 0)}{P(t = t_1t_3t_3|s = 1)} = \left(\frac{1-\alpha}{\alpha}\right)^l \left(\frac{\alpha}{1-\alpha}\right)^{3-l} = \left(\frac{1-\alpha}{\alpha}\right)^{2l-3}$$

Comme $\alpha < 0.5$ la quantité $\dfrac{1-\alpha}{\alpha} > 1$ est supérieure à 1. Nous remarquons alors que si les 0 sont majoritaires dans t, c'est à dire si $2l > 3$, la probabilité de réaliser une telle séquence t, sachant que le mot envoyé est 000 est plus importante que la probabilité de réaliser la même séquence en envoyant le mot 111. D'après la règle de vote majoritaire dans ce cas on choisit justement le mot 000. Si ce sont les 1 qui sont majoritaires alors $2l - 3 < 0$ et on choisit encore des deux hypothèses celle qui rend la séquence de bits reçue la plus probable. C'est ce qu'on appelle règle de maximum de vraisemblance.

2.3 Groupe G_k.

Considérons l'ensemble de séquences binaires de taille k. On le note $G_k = \{0,1\}^k$. On a $Card(G_k) = 2^k$. On pourra appeler les éléments de cet ensemble mots ou vecteurs. Sur cet ensemble nous définissons l'opération d'addition modulo 2, notée $\oplus$, entre les éléments. Cela correspond à effectuer un XOR (un ou exclusif) bit par bit, sachant que

b_1	b_2	$b_1 \oplus b_2$
1	1	0
1	0	1
0	1	1
0	0	0

Muni de cette loi additive, G_k est un groupe abélien d'élément neutre $(0,\ldots,0)$. Si on le munit également de loi externe multiplicative selon la table de ET logique :

b_1	b_2	$b_1 \otimes b_2$
1	1	1
1	0	0
0	1	0
0	0	0

G_k est un espace vectoriel.

2.4 Distance de Hamming et décodage du maximum de vraisemblance

2.4.1 Distance de Hamming

On définit sur G_k une distance de façon suivante.

Définition 6.7 (Distance de Hamming)**.** Soient $v = (v_1,\ldots,v_k),\ w = (w_1,\ldots,w_k) \in G_k$ deux vecteurs de G_k. On appelle distance de Hamming le nombre de bits différents entre v et w

$$d_h(v,w) = Card\{i \in [1,\ldots,n] \mid v_i \neq w_i\}$$

Il est assez facile de vérifier que la distance de Hamming est une distance, c'est à dire qu'elle vérifie les trois axiomes :

1. $\forall\ w,v \in G_k, d(v,w) \geq 0$ et $d(w,v) = 0 \quad \Leftrightarrow \quad w = v$
2. $d(v,w) = d(w,v), \quad \forall v,w \in\ G_k$
3. $d(w,v) \leq d(w,z) + d(z,v), \quad \forall w,v,z \in G_k$

A ce titre, la distance de Hamming induit une certaine topologie sur l'ensemble G_k. En particulier, on peut définir la notion de boule ouverte de rayon r centrée en $w \in G_k$ de façon suivante :

$$B_h(w,r) = \{v \in G_k, \quad d_h(w,v) < r\} \tag{6.1}$$

et de boule fermée :

$$\overline{B_h}(w,r) = \{v \in G_k, \quad d_h(w,v) \leq r\} \tag{6.2}$$

Une boule ouverte $B_h(w,r)$ définit l'ensemble de mots binaires $v \in G_k$ qui diffèrent de w en moins de r bits. Par exemple, si $k = 4$ et $w = 1010$ alors

$$B_h(w,2) = \{1010, 0010, 1110, 1000, 1010\}$$

C'est l'ensemble de mots que l'on peut obtenir à partir de w en changeant 0 ou 1 bits à la fois.

Définition 6.8 (Poids d'un vecteur)**.** Soit $v = (v_1, \ldots, v_k) \in G_k$. On appelle poids de v le nombre de bits différents de 0 dans v :

$$\omega(v) = Card\{i \in [1, \ldots, n] \mid v_i = 1\}$$

On peut montrer facilement que

Proposition 6.1. *Soient $v, w \in G_k$. Alors*

$$d_h(v,w) = \omega(v \oplus w)$$

Exemple 6.1. Soit $k = 5$. G_5 contient 2^k mots binaires de longueur 5. Soient

$$v = 00110, \quad w = 10101$$

On a alors

$$d_h(w,v) = 3, \quad w \oplus v = 10011, \quad \omega(w \oplus v) = 3$$

Puisque la distance de Hamming comptabilise les différences entre deux mots binaires, il est naturel de l'utiliser comme critère de décodage après une transmission erronée.

Définition 6.9 (Décodage selon le minimum de distance de Hamming)**.** Soit une suite de mots $C = \{w_1, \ldots, w_M\} \subset G_n$ de longueur n qui forment n code. Soit $z \in G_n$ un mot de longueur n reçu après transmission à travers d'un canal bruité d'un des mots du code C. la règle de décodage selon le minimum de distance de Hamming associe à z le mot du code w le plus proche de z selon d_h :

$$g(z) = w_0 \in C \quad \text{t.q.} \quad d_h(z, w_0) = \min_{w \in C} d_h(z,w)$$

2.4.2 Décodage au sens de maximum de vraisemblance

Définition 6.10. Soit une suite de mots $C = \{w_1, \ldots, w_M\} \subset G_n$ de longueur n qui forment n code. Soit $z \in G_n$ un mot de longueur n reçu après transmission à travers d'un canal bruité d'un des mots du code C. On appelle décodage au sens de maximum de vraisemblance la règle de décodage qui associe à z le mot $w \in C$ qui maximise la probabilité de réalisation de z sachant w :

$$g(z) = w_0 \in C \quad \text{t.q.} \quad P[z|w_0] = \max_{w \in C} P[z|w]$$

Nous allons montrer maintenant une proposition importante qui établit le lien entre le décodage de maximum de vraisemblance et la distance de Hamming.

Proposition 6.2. *Si le canal de transmission est binaire et symétrique alors le décodage de maximum de vraisemblance est équivalent au décodage au sens de minimum de distance de Hamming.*

Preuve la Proposition 6.2

Supposons que nous utilisons un canal binaire symétrique de probabilité d'erreur $\alpha < 0.5$. Soit $z = z_1 z_2 \ldots z_n$ le mots binaire reçu. Rappelons que la transmission de chaque symbole est indépendante des autres. Soit un mot de code $w \in C$. Admettons l'hypothèse que on a reçu z en ayant envoyé w. Alors chaque bit différent entre z et w représente une erreur de transmission, de probabilité $\alpha < 0.5$. Tous les autres bits sont des transmissions corrects de probabilité $1 - \alpha$. Le nombre de bits différents entre z et w correspond à la distance de Hamming entre les deux mots : $d_h(z, w)$. Alors on a

$$P[z|w] = \alpha^{d_h(z,w)}(1-\alpha)^{n-d_h(z,w)}$$

Soient deux mots de code w_1 et w_2. Notons $d_1 = d_h(z, w_1), \quad d_2 = d_h(z, w_2)$. Comparons les vraisemblances associées aux deux mots :

$$\frac{P[z|w_1]}{P[z|w_2]} = \frac{\alpha^{d_1}(1-\alpha)^{n-d_1}}{\alpha^{d_2}(1-\alpha)^{n-d_2}} = \left(\frac{1-\alpha}{\alpha}\right)^{d_2-d_1}$$

Comme $\alpha < 0.5$ on a $\dfrac{1-\alpha}{\alpha} > 1$. Alors d'après la dernière relation

$$P[z|w_1] \leq P[z|w_2] \quad \Leftrightarrow \quad d1 \geq d2$$

Ainsi, la vraisemblance $P[z|w]$ est **maximale** quand la distance de Hamming $d_h(z, w)$ est **minimale. C.Q.F.D**

Dans la suite nous allons travailler avec la règle de décodage du minimum de distance de Hamming.

2.5 Codes correcteurs

Définition 6.11 (Code (k, n))**.**
- On appelle **code de paramètres** (k, n) toute application injective $\phi : G_k \to G_n$.
- Le paramètre k s'appelle **dimension du code**. Le paramètre n s'appelle **longueur du code**.
- L'ensemble image de ϕ,
$$C = \{\phi(m), \quad m \in G_k\}$$
s'appelle **image de code**. Ses éléments s'appellent **mots de code**. Les éléments de G_k s'appellent **mots de source**.

Un code est une association entre les blocks de longueur k et des mots binaires de longueur n. On demande que l'application qui le représente soit injective pour garantir que deux blocks différents soient codés par deux mots différents.

Exemple 6.2. Le code de répétition pure vu ci-dessus est un code de paramètres $(1, 3)$. Les mots de source sont $\{m_1 = 0, m_2 = 1\}$. L'image de ce code est
$$C = \{000, 111\}$$

Définition 6.12 (Distance minimale d'un code)**.** Soit un code de paramètres (k, n) dont l'image est l'ensemble de mots C. On appelle distance minimale d'un code la valeur
$$d = min_{w,v \in C} d_h(w, v)$$
c'est la plus petite distance de Hamming entre deux mots du code.

Ce paramètre joue un rôle très important dans la définition des capacités d'un code de détecter et corriger les erreurs de transmission.

2.5.1 Détection et correction d'erreurs

On suppose que la règle de décodage utilisée est celle de minimum de distance de Hamming. Soit un code (k, n) d'image C. Supposons qu'un mot du code $w \in C$ est transmis. On reçoit le mot $z \in G_n$. Si z coïncide avec un des mots de code, $v \in C$, il sera décodé comme v. Même si ce n'est pas le mot du code qui a été transmis. Cela peut se produire si les erreurs se produisent exactement sur les bits qui sont différents entre le mot transmis, w et le mot du code reçu, v. Donc s'il y a $d_h(w, v)$ erreurs et si elles se produisent "aux bons endroits", elles sont indétectables. On détecte les erreurs si le mot reçu, z ne correspond à aucun des mots de code.

Ainsi nous pouvons dire que le nombre maximal d'erreurs qu'il est possible de détecter quels que soient les bits où elles se produisent est égal à $d - 1$, où d est la distance minimale du code. En effet, pour obtenir à partir d'un mot du code $w \in C$ un autre mot du code, $v \in C$ il faut changer au moins d bits. Donc si on change moins de d bits on obtient forcément un mot qui n'est pas dans le code. Donc on détecte la présence des erreurs à la réception. Nous venons de justifier la définition suivante :

Définition 6.13 (Capacité de détection)**.** On appelle capacité de détection d'erreurs d'un code le nombre maximal e_d d'erreurs qu'il est **toujours** possible de détecter. On a

$$e_d = d - 1$$

où d est la distance minimale du code.

Lorsque les erreurs sont détectées on ne sait pas quels sont les bits erronés et combien sont ils. On sait juste que le mot reçu n'est égal à aucun des mots du code. La règle de décodage que l'on a choisi va alors remplacer le mot reçu par celui des mots du code qui est le plus proche selon la distance de Hamming. Si cette action conduit au mot qui a réellement été transmis, elle équivaut à la correction des erreurs. Nous allons donc définir un autre paramètre important pour juger de la qualité d'un code :

Définition 6.14 (Capacité de correction)**.** On appelle capacité de correction d'erreurs d'un code le nombre maximal e_c d'erreurs qu'il est **toujours** possible de corriger. On a

$$e_c = E\left[\frac{d-1}{2}\right]$$

où d est la distance minimale du code, $E[]$ est la partie entière d'un réel.

Preuve La formule donnée dans la définition de la capacité de correction nécessite une justification. Remarquons que si $C = \{w_1, \ldots, w_M\}$ est l'image d'un code (k, n) de distance minimale d alors toutes les boules fermées de rayon $r = E\left[\frac{d-1}{2}\right]$ (voir (6.2)) $\overline{B_h}(w_j, r)$ sont deux à deux disjointes. Soit $w \in C$ le mot de code transmis. tant que le mot reçu reste dans la boule $\overline{B_h}(w, r)$ il sera décodé comme w et donc les erreurs seront corrigées. Ainsi le plus grand nombre d'erreurs corrigibles et $E\left[\frac{d-1}{2}\right]$.

2.6 Codes linéaires correcteurs d'erreurs

Nous allons enfin nous intéresser ici d'une famille particulière de codes (n, k), les codes linéaires. Leur principal intérêt dans la facilité de mise en oeuvre et la rapidité des algorithmes de décodage.

Définition 6.15 (Code correcteur linéaire (n, k))**.** Un code linéaire (n, k) est un code de classe (n, k) tel que l'application $\phi : G_k \to G_n$ qui le génère est une application linéaire, c'est-à-dire, tel qu'il existe une matrice $G \in \mathcal{M}^{n,k}(0, 1)$ telle que

$$\phi(m) = Gm, \quad \forall m = \begin{pmatrix} m_1 \\ \vdots \\ m_m \end{pmatrix}$$

On appelle G la matrice génératrice du code.

Remarque 6.1. Ici les coefficients de la matrice G prennent les valeurs dans $\{0,1\}$. La multiplication matricielle se fait selon les opérations $\oplus$, $\otimes$ sur $\{0,1\}$, c'est à dire dans l'arithmétique modulo 2. Par exemple,

$$\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix} = \begin{pmatrix}(1\otimes 1)\oplus(1\otimes 1)\\ (0\otimes 1)\oplus(1\otimes 1)\\ (1\otimes 1)\oplus(0\otimes 1)\end{pmatrix} = \begin{pmatrix}0\\ 1\\ 1\end{pmatrix}$$

Remarque 6.2. Si le code est linéaire alors son image, C, l'ensemble de tous les mots de code, est un sous-espace vectoriel de G_n. Il contient donc obligatoirement le mot nul $000\ldots000$.

Définition 6.16. On dit qu'un code linéaire (k,n) est systématique ssi sa matrice génératrice G est de la forme (en blocks)

$$G = \begin{pmatrix}I_k\\ G'\end{pmatrix}$$

où I_k est la matrice identité de dimension k et G' est une matrice de $n-k$ lignes et k colonnes.

Proposition 6.3. *Les codes systématiques ont la propriété intéressante : pour tout mot de source $m \in G_k$ le mot de code correspondant $w = Gm$ a m comme préfixe. C'est à dire*

$$w = mw'$$

Exemple 6.3. Un code de parité simple consiste à ajouter à chaque mot de source un bit qui représente la parité du nombre de 1 dans le block. Si le nombre de 1 est pair on ajoute 0 sinon, on ajoute 1. Par exemple, pour les blocks de 3 bits le code de parité est défini par le tableau suivant :

m	000	100	010	001	110	011	101	111
$\phi(m)$	0000	0001	0101	0011	1100	0110	1010	1111

C'est un code de paramètres $(3,4)$. Chaque mot de source m est le préfixe du mot de code correspondant. C'est un code systématique. Il est linéaire car on peut remarquer que le bit de parité est toujours égal à la somme (modulo 2 !) des bits du mot m. Alors la matrice génératrice de ce code est très simple :

$$G = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 1 & 1\end{pmatrix}$$

Définition 6.17. Deux codes sont dits **équivalents** s'ils ont la même image.

Pour déterminer si un code linéaire de matrice génératrice donnée possède un équivalent systématique il suffit d'effectuer une opération d'échelonnage de la matrice en faisant des opérations **élémentaires sur les colonnes**.

Exemple 6.4. Soit le code de matrice

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Notons les colonnes C_1, C_2, C_3. Les premières opérations visent à annuler les éléments G_{11} et G_{13}. Le pivot est G_{11}. On pose $C_1 \leftarrow C_1,\ C_2 \leftarrow C_1 + C_2,\ \ C_3 \leftarrow C_3$.

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \begin{vmatrix} C_1 \leftarrow C_1 \\ C_2 \leftarrow C_2 \\ C_3 \leftarrow C_3 + C_2 \end{vmatrix} \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \begin{vmatrix} C_1 \leftarrow C_1 + C_3 \\ C_2 \leftarrow C_2 + C_3 \\ C_3 \leftarrow C_3 \end{vmatrix} \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

On obtient ainsi un équivalent systématique de notre code car les trois première lignes de la matrice G forment maintenant une matrice identité.

La proposition suivante donne ne méthode très simple pour évaluer la distance minimale (et donc les capacités de détection et de correction d'erreurs !) d'un code linéaire.

Proposition 6.4 (Distance minimale d'un code linéaire)**.** *La distance minimale d'un code linéaire est égale au plus petit poids non nul des mots de code.*

Preuve la Proposition 6.4

Rappelons que la distance minimale d'un code est

$$d = \min_{w,v \in C} d_h(w, v)$$

remarquons également que

$$d_h(w, v) = \omega(w \oplus v)$$

Or, si le code est linéaire, son image C est un sous-espace vectoriel. Donc pour tout $w, v \in C$ on a $w + v \in C$. Si $w \neq v$ alors $w \oplus v \neq 0$. Donc

$$d = \min_{w,v \in C,\ w \neq v} d_h(w, v) = \min_{z \in C,\ z \neq 0} \omega(z)$$

C.Q.F.D

Proposition 6.5 (Borne de Singleton)**.** *La distance minimale d'un code linéaire de dimension k et de longueur n vérifie*

$$d \leq n + 1 - k$$

2.6.1 Matrice de contrôle et décodage d'un code linéaire

Définition 6.18 (Matrice de contrôle d'un code linéaire)**.** Soit un code linéaire (k, n) de matrice génératrice G et d'image C. On appelle matrice de contrôle d ce code toute matrice $H \in \mathcal{M}_{n-k,n}$ telle que pour tout mot de code $w = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ on ait

$$Hw = 0$$

Autrement dit, H est une matrice dont le noyau coïncide avec l'ensemble de mots de code C.

Généralement il n'est pas facile de trouver la matrice de contrôle pour un code donné. Du point de vue algébrique, cela revient à trouver la base du sous-espace vectoriel orthogonal à C. Mais dans le cas particulier de codes systématiques la proposition suivante permet de construire facilement la matrice de contrôle :

Proposition 6.6 (Matrice de contrôle d'un code systématique)**.** *Soit un code linéaire* (k, n) *systématique de matrice génératrice* $G = \begin{pmatrix} I_k \\ G' \end{pmatrix}$ *Alors sa matrice de contrôle est*

$$H = (G' I_{n-k})$$

Définition 6.19 (Syndrome et erreur)**.** Soit un code linéaire (k, n) de matrice génératrice G et de matrice de contrôle H. Soit $Y \in C$ un mot de code et $Z \in \{0, 1\}^n$ un mot reçu après la transmission de Y par un canal bruité.

On appelle **mot erreur** le mot $E \in \{0, 1\}^n$ tel que

$$Z = Y + E$$

On appelle **le mot syndrome** de Z

$$S = HZ$$

Chaque bit de mot erreur E est nul si les bits correspondants de Z et Y coïncident, et est égal à 1 s'ils sont différents. Ainsi

$$\omega(E) = d_h(Y, Z)$$

Si on a trouvé le mot erreur E on peut retrouver Y en posant :

$$Y = Z + E$$

Donc, pour corriger les erreurs de transmission, il suffit de trouver le vecteur E.

Si le syndrome de Z $S = HZ$ est nul c'est que Z est un mot de code (par définition de la matrice de contrôle). Il sera alors décodé par le block correspondant dans la table du code. Ainsi, **le syndrome nul correspond à l'absence ou à la non détection des erreurs de transmission.**

Le syndrome non nul équivaut la détection d'erreurs, car cela signifie que le mot reçu Z n'est pas un mot de code.

Proposition 6.7. *Dans les hypothèses de la définition 6.19, soient Y le mot transmis et Z le mot reçu. Soient $E = Z + Y$ le mot erreur et $S = HZ$ le syndrome de Z. Alors le mot erreur a le même syndrome que Z : $HE = HZ$*

Preuve la Proposition 6.7

Par définition de la matrice de contrôle on a

$$HY = 0$$

car Y est un mot de code. Alors

$$HE = H(Y + Z) = HZ = S$$

C.Q.F.D

Définition 6.20 (Classe latérale)**.** Soit un code linéaire (k, n) de matrice de contrôle H. Soit Z un mot de $\{0, 1\}^n$. On appelle classe latérale de Z l'ensemble de mots ayant le même syndrome que Z :

$$L(Z) = \{W \in \{0, 1\}^n, \quad HW = WZ = S\}$$

Ainsi nous savons où chercher le vecteur erreur E pour la correction du mot reçu, Z. Il se trouve dans la classe latérale de Z. Il reste à savoir comment le choisir. Rappelons que le décodage se fera selon le principe de minimum de distance de hamming. Le mot reçu Z sera remplacé par le mot de code W le plus proche. Nous avons déjà mentionné que le poids du vecteur erreur $E = Z + W$ entre le mot reçu Z et un mot de code W est égal à la distance de Hamming entre Z et W. Or, d'après la proposition 6.7, on a

$$\forall W \in C, \quad E_W = Z + W \in L(Z)$$

Et réciproquement,

$$\forall E \in L(Z), \quad W = Z + E \in C$$

Donc trouver le mot de code le plus proche du mot reçu Z équivaut à trouver l'élément $E \in L(Z)$ de plus petit poids.

Proposition 6.8. *Soit un code linéaire de matrice de contrôle H. Le décodage au sens de minimum de distance de hamming d'un mot Z revient à trouver dans la classe latérale de Z le mot de plus petit poids, E_0 et à remplacer Z par le mot de code $W = Z + E \in C$.*

En pratique, on explore $L(Z)$ dans l'ordre croissant des poids de mots. Pour faciliter les explications nous allons introduire une famille de mots particuliers :

$$E_i = (00 \dots 010 \dots 00)$$

où pour chaque mot E_i, $i = 1, \ldots, n$ il n'y a qu'un seul bit égal à 1, à la i-ème position. Remarquons un mot E de poids p est égale à la somme de p mots parmi les E_i, $i = 1, \ldots, n$

$$\omega(E) = p \quad \Leftrightarrow \quad \exists i_1, \ldots, i_p, \quad t.q. \quad E = \sum_{j=1}^{p} E_{i_j}$$

Alors le résultat de multiplication HE est une combinaison linéaire des colonnes de la matrice H, $C_1, \ldots, C_n$:

$$HE = \sum_{j=1}^{p} C_{i_j}$$

car les autres bits de E sont nuls.

Revenons maintenant au problème de recherche de l'erreur de poids minimum dans la classe latérale d'un mot reçu Z. Soit $S = HZ$ le syndrome de Z. Soit $0 \leq p \leq e_c$. (e_c est la capacité de correction du code). Dire qu'il existe dans $L(Z)$ un mot E de poids p équivaut à

$$\exists i_1, \ldots, i_p, \quad t.q. \quad E = \sum_{j=1}^{p} E_{i_j} \quad \text{et} \quad HE = \sum_{j=1}^{p} C_{i_j} = S$$

Autrement dit, il existe un choix de p colonnes de la matrice de contrôle H dont la somme est égale à S. Alors on commence teste les poids p allant de 0 à e_c jusqu'à ce qu'on trouve une somme de p colonnes de H égale à S. Le vecteur erreur est alors égal à la somme correspondante de E_{i_j}.

Par exemple, pour $p = 0$. Si on trouve que $HZ = 0$ alors Z est un mot de code. Sinon, on passe à $p = 1$. Pour $p = 1$ si on trouve une colonne C_i de H telle que $C_i = S$ alors le mot de code corrigé sera $W = Z + E_i$. Sinon on passe à $p = 1$. S'il existe deux colonnes parmi n de la matrice H telles que $C_i + C_j = S$ alors le mot corrigé sera $W = Z + E_i + E_j$. Sinon, on continue avec $p = 3$ et ainsi de suite. On a donc l'algorithme suivant :

Algorithme 6.1 (Décodage d'un code linéaire)**.** .

Etape 1. $p = 0$. *On vérifie si* $S = HZ = 0$. *Si oui,* Z *est un mot de code. Sinon, on pose* $p = 1$ *et on passe à l'étape 2.*

Etape 2 *Tant que* $p \leq e_c$ *faire*

tester toutes les sommes de p *colonnes de* H *jusqu'à ce que*

$$\sum_{j=1}^{p} C_{i_j} = S$$

Si trouvé, renvoyer $W = Z + \sum_{j=1}^{p} E_{i_j}$. *Sinon, poser p=p+1.*

Fin de tantque.

Etape 3 *Si* $p = e_c + 1$ *et aucun solution n'est trouvée, renvoyer "erreur de décodage".*

Index