Comme $D\left[p(x) \| q(x)\right]$ est positif $H(X) \leq \log_2 N$ ce que l'on connaissait par ailleurs.

Transmission d'information

Ce chapitre identifie les problèmes à résoudre lors de la transmission d'un message d'un émetteur vers un destinataire en s'appuyant sur des exemples de la vie courante.

Depuis que l'homme sait parler et écrire, il a toujours essayé de communiquer avec ses semblables. C'est grâce à la possibilité d'échanger de l'information mais aussi de la stocker que les civilisations humaines ont pu progresser et atteindre le niveau avancé d'aujourd'hui. Pendant la majeure partie de l'histoire humaine, l'information s'est transmise sous forme analogique (parole, écriture). Grâce aux ordinateurs, le traitement numérique de l'information a pris aujourd'hui une place majeure dans la vie courante. Ne pas savoir utiliser un ordinateur de nos jours est devenu un handicap important au même titre que de ne pas savoir conduire une voiture ce qui, dans ce dernier cas, restreint fortement les possibilités de se déplacer si l'on habite dans une région dépourvue de transports en communs. De même qu'il n'est pas nécessaire de connaître la technologie d'une voiture pour pouvoir la conduire, il n'est pas nécessaire de connaître celle d'un ordinateur pour s'en servir.

1. Transmettre de l'information

Transmettre de l'information entre deux individus éloignés n'est pas un problème facile. Voyons cela sur quelques exemples de la vie courante.

Prenons le cas d'un français, Monsieur Martin ne parlant pas chinois, qui veut transmettre un message à Monsieur Chen ne parlant que le chinois, chacun étant dans son pays d'origine. Comment va-t-il faire ?

Si Monsieur Martin envoie une lettre en français, Monsieur Chen ne pourra pas la comprendre sauf s'il y a un interprète. De même, si Monsieur Martin téléphone à Monsieur Chen, ils ne se comprendront pas. Dans une vidéoconférence entre ces deux personnes, ils pourront échanger un peu d'information grâce à des gestes. On voit donc que l'échange d'information entre un émetteur et une source est impossible s'ils ne parlent pas le même langage. Pour échanger de l'information de manière efficace, Messieurs Martin et Chen auront besoin d'avoir une langue commune ou il leur faudra utiliser un interprète qui n'est autre qu'un décodeur de leur langue vers la langue de leur interlocuteur.

Alice et Véronique parlent la même langue. Elles sont à trois mètres l'une de l'autre au réfectoire de l'université. Elle se parlent et échangent donc de l'information. Toutefois le bruit fait par les autres étudiants avec les couverts et leurs conversations est tel qu'elles ne comprennent pas toujours tout ce qu'elles disent. Elles doivent se répéter et parfois ce que dit Véronique est mal compris par Alice.

Alice écoute le cours du professeur et prend des notes écrites. Elle code avec des lettres les sons qu'elle entend. Après le cours elle tapera le texte qu'elle a écrit sur son ordinateur et enverra le fichier par mail à son amie Véronique qui n'a pas pu venir au cours. Ce texte sera codé en binaire dans l'ordinateur, transmis ainsi via le réseau et décodé par l'ordinateur de Véronique qui verra apparaître sur son écran le texte envoyé par Alice. Véronique imprimera celui-ci sur du papier avec son imprimante.

Après le cours, Alice et Véronique sortent de l'amphithéâtre. Alice a un secret à confier à Véronique. Elle va lui parler dans le creux de l'oreille et lui tendre un petit papier plié en quatre dans lequel elle a mis des informations qu'elle souhaite cacher aux autres étudiants. Alice transmet des informations confidentielles à Véronique. Mais elle n'a pas de chance, Pierre, un autre étudiant, qui aime bien rapporter les potins, a vu qu'Alice et Véronique se parlaient à l'oreille et qu'Alice a donné un petit bout de papier que Véronique a mis dans sa poche. Il s'approche, bouscule un peu Véronique en s'excusant et subtilise de ce petit papier.

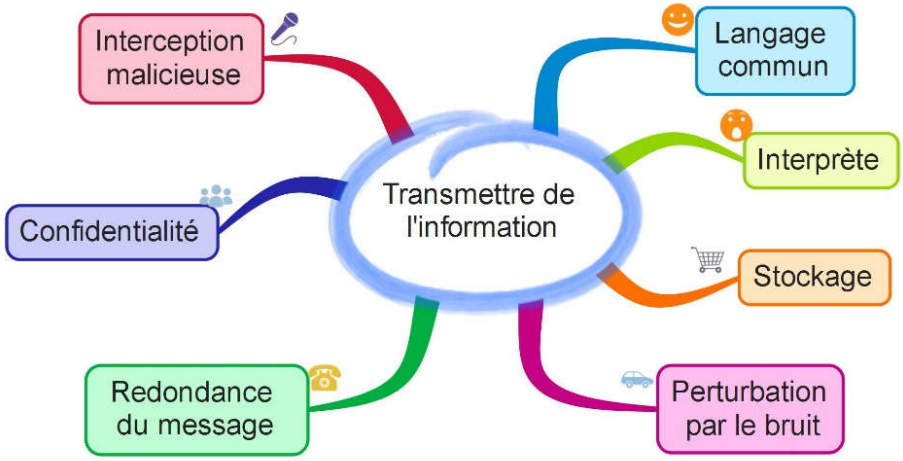

Figure 1

Jean doit donner son nouveau numéro de téléphone à Jérémy mais il y a de la friture sur la ligne. Il répète donc deux fois son numéro de téléphone et demande à Jérémy qui l'a noté sur un bout de papier de le lui répéter pour contrôler qu'il ne s'est pas trompé.

Alice a eu un accrochage avec sa voiture. Elle veut faire sa déclaration sur internet mais l'assurance ne l'autorise qu'à utiliser que 256 caractères, blancs compris. Dans un premier jet qu'elle a fait sur le papier, il lui a fallu 500 caractères pour décrire les circonstances de l'accident et les dégâts. Il faut qu'elle réduise (comprime) son texte pour qu'il tienne dans 256 caractères.

Dans toutes ces situations de la vie quotidienne il y a eu un transfert d'information d'un individu à un autre ou d'un point à un autre. Ces exemples permettent de faire émerger plusieurs propriétés que doivent posséder des systèmes de transfert d'information

Pour transmettre de l'information il faut considérer un certain nombre de choses afin que celle-ci soit fidèle et fiable (figure 1). Parfois le message reçu doit être exactement égal au message émis. Pour d'autres applications ce n'est pas nécessaire comme lors de la transmission d'images, de musique ou de vidéos. Pour ces dernières il suffit que les distorsions du signal originale soient très faibles. C'est par exemple le cas de la musique codée mp3 où il y a perte d'information par rapport à la musique initiale ou des images codées en jpeg ou jpg. Dans le cas des images, par exemple, si l'on travaille sur une image jpg avec un

logiciel de retouche d'image et qu'on l'enregistre régulièrement le fichier correspondant, les petites pertes lors de chaque enregistrement s'accumulent et peuvent au total, si le nombre d'enregistrement est grand, conduire à une modification visible de l'image initiale.

Pour transmettre de l'information (figure 1) il faut un langage commun sinon on doit recourir aux services d'un interprète. On peut avoir besoin de la stocker, en écrivant un message sur un bout de papier, par exemple. Lors de la transmission du message, des perturbations peuvent l'altérer d'où la nécessité d'une certaine redondance (on répète un numéro de téléphone). Une personne étrangère peut même intercepter une conversation si bien qu'une confidentialité des échanges est nécessaire.

2. Pourquoi coder ?

Le codage (ou décodage) d'une information est une étape essentielle si l'on veut transmettre de l'information d'un émetteur vers un destinataire. Celui a trois fonctions principales (figure 2) :

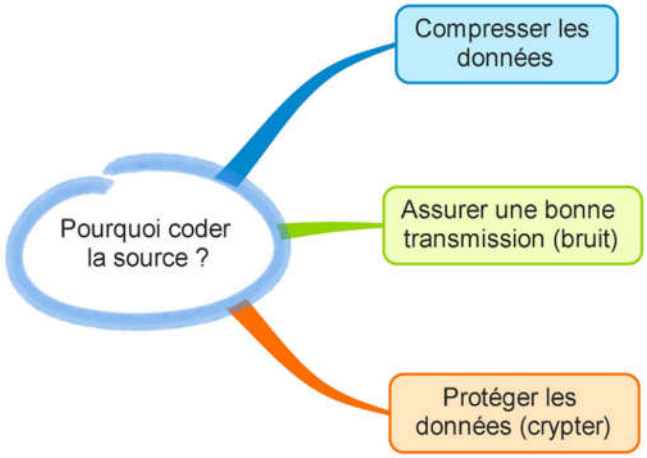

Figure 2

1. Il doit permettre de compresser les données car le transfert de celles-ci se fait dans des canaux qui ont un débit limité. C'est un peu comme une canalisation dans laquelle on veut faire circuler de l'eau. Elle permet un débit maximum. Le but est donc de réduire la longueur du message initial pour qu'il occupe le moins de place possible. Cela correspond à la phase de compression du message. Lors de cette étape, on réduit aux maximum les redondances existantes.
2. La transmission de l'information est toujours perturbée par des parasites et autres perturbations qui font qu'elle n'arrive pas intacte. Elle peut être déformée ou perdue. Il faut tenir compte de ces perturbations, corriger les erreurs ou savoir que le message n'est pas intègre pour le transmettre à nouveau. Cette étape a toujours pour conséquence d'augmenter la taille du message initialement comprimé.
3. La plupart des informations que l'on transmet doivent être protégées car elles peuvent être interceptées, modifiées, etc. Lorsque l'on fait un achat sur internet avec sa carte bleue, il ne faut pas que les informations de la carte soient interceptées par un pirate qui pourrait ensuite s'en servir de manière frauduleuse. La cryptographie joue un rôle important dans la transmission d'information.

3. De la source au destinataire

La figure 3 montre le schéma d'une chaîne de transmission de l'information.

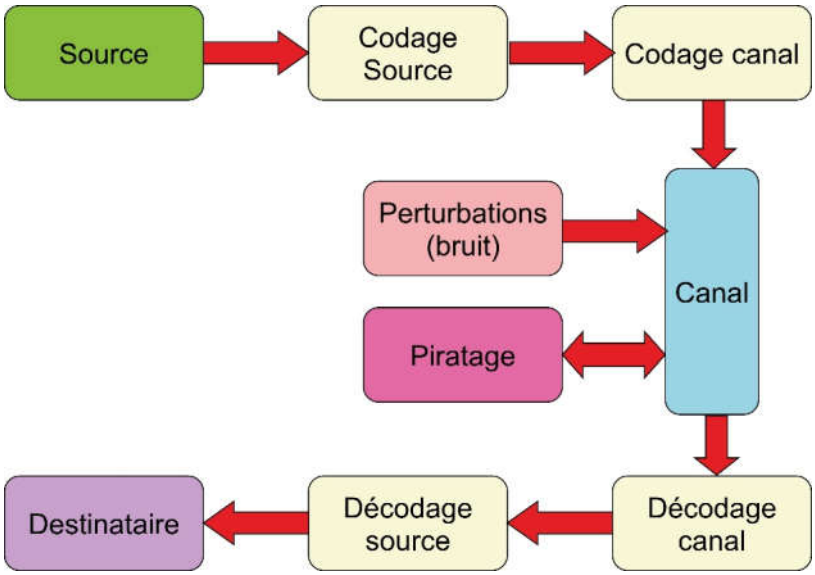

Figure 3

Le point de départ est une source d'information qui peut être un message alphanumérique écrit à la main ou un concert musical dans un auditorium. La première chose à faire est de coder les signaux issus de cette source, en les tapant sur un clavier d'ordinateur pour le premier cas ou en enregistrant la musique en analogique ou digital dans le second.

Dans tous les cas il est préférable d'obtenir une information digitale pour pouvoir la transmettre avec les réseaux actuels. Il est en effet beaucoup plus difficile de corriger des erreurs lors d'une transmission analogique.

Comme le canal de transmission a une capacité limitée, il faut réduire au maximum la place occupée par le message tout en ne perdant pas d'information (cas d'un message écrit) ou en perdant le moins possible (cas de la musique, vidéo). C'est le rôle du codage source dans lequel on prépare l'information à transmettre pour qu'elle occupe le moins de place possible. C'est à ce stade que l'on comprime l'information.

Malheureusement lors de la transmission dans le canal, des perturbations peuvent modifier le signal ou le rendre illisible en sortie de canal. Avant d'injecter l'information dans le canal, il faut rajouter de la redondance au message initial afin de pouvoir détecter des erreurs de transmission pour les corriger quand c'est possible ou éventuellement les corriger ou demander une retransmission du message. Ajouter de la redondance veut dire qu'on va augmenter la quantité d'information à transmettre.

Les perturbations dans le canal peuvent être stochastiques et involontaires quand elles proviennent du bruit électronique ambient ou de signaux parasites. Elles peuvent aussi être volontaires lorsqu'il s'agit de piratage du canal. C'est la raison pour laquelle il faut aussi crypter l'information véhiculée dans le canal afin d'en protéger le contenu.

En sortie de canal, le message doit être décodé et décrypté et transformé en une formule compréhensible par le destinataire.

Codage source

Avant d'être transmis, un message doit être codé pour occuper le moins de place possible. Ce codage de la source doit être le plus efficace possible et rapide à mettre en œuvre. Dans ce chapitre nous introduisons le sujet et présentons quelques exemples de codage.

Une source d'information génère des messages qui sont constitués d'une séquence de symboles appartenant à un alphabet. Nous ne considérons ici que des alphabets discrets. Ainsi, un texte écrit, comme l'article de journal, est un message. Il est constitué de lettres pris dans les 26 lettres de l'alphabet (majuscules et minuscules) auquel on ajoute les chiffres et certains symboles utiles à la compréhension du texte (point, virgule, point d'interrogation, etc.).

Un autre exemple de message est une image constituée de pixels. Ceux-ci seront d'autant plus nombreux que la résolution de l'image sera importante. Une image en couleur occupera beaucoup plus de mémoire qu'une image en noir et blanc ou avec des niveaux de gris.

1. Codage

Soit une source d'information émettant un message S_t au temps (figure 1). Ce message est transformé en un ensemble de **mots de code** C_t par le codeur. S_t est construit à partir d'un alphabet A_{Source} qui est par exemple constitué des lettres de l'alphabet dans le cas d'un message alphanumérique[8]. C_t est construit à partir d'un alphabet A_{Codeur} qui dans le traitement digital de l'information est l'alphabet binaire : 0 ou 1. Un symbole du message source (par exemple une lettre) donne un mot de code lorsqu'il est codé en binaire qui contient en général plusieurs symboles de l'alphabet A_{Codeur}. Par exemple, le symbole « A » dans l'alphabet source est transformé en $41_{16} = 1000001_2$ si on le code en ASCII.

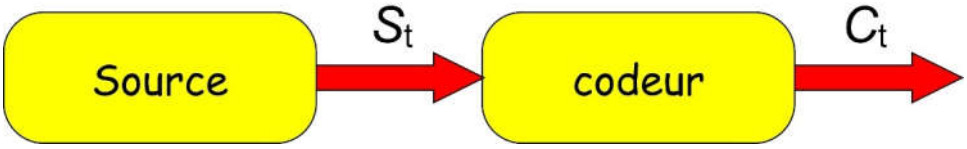

Figure 1

Pour transmettre un message, on a tout intérêt à ce qu'il occupe le moins de place possible. Pour cela, on peut utiliser plusieurs possibilités de codage plus ou moins performantes. Illustrons cela sur deux exemples.

[8] La taille de l'alphabet est appelée **arité** de la source.

Considérons une feuille de papier A4 (21×29,7 cm) dont on souhaite transmettre un seul côté par fax avec un modem dont le débit est de 33,6 kbits/seconde (kbps). On veut une résolution de 300 dpi (dots per inch) et une transmission en noir et blanc. Chaque point est donc un bit dont la valeur peut être 0 (blanc, par exemple) ou 1 (noir). La quantité d'information à transmettre vaut $\frac{21\times 29,7}{(2,54)^2}\times 300^2 \Rightarrow 8,7 \text{ Mbits}$. Avec un modem de 33,6 kbps, il faut 259 secondes, soit 4 minutes et demi pour transmettre cette page. Si le fax émettait 24 heures sur 24, on ne pourrait transmettre que 333 pages ce qui est peu dans le cadre d'une entreprise. Il est donc essentiel de faire un codage qui comprime l'information initiale. En utilisant un codage pour réduire la taille du message, on peut transférer une page en moins de 20 secondes ce qui est 10 fois plus rapide que sans compression.

La domaine de la musique illustre l'évolution des méthodes de compression selon le choix de l'algorithme de codage. Si l'on grave un CD en PCM (Pulse Code Modulation (WAV sous windows et aiff sous Mac Os), la résolution est de 16 bits et la fréquence d'échantillonnage de 44,1 kHz. Pour un morceau en stéréo de 3 minutes 30, il faut un débit de $44,1\times 16\times 2 = 1411 \text{ kb/s}$. La taille du morceau est alors de $\frac{44,1\times\frac{16}{8}\times 210\times 2}{1024} = 36,2 \text{ Mo}$. On peut mettre un peu moins d'une vingtaine de chansons de variété sur un CD.

Le codage mp3 (Moving Picture Expert Group 1 layer 3) comprime plus efficacement un morceau de musique au prix d'une perte de qualité minime. Avec l'algorithme de codage à 128 kb/s (bitrate ou nombre de bits autorisés par seconde) 1 mn de musique stéréo occupe $\frac{128\times 10^3\times 60}{8} \simeq 1 \text{ Mo}$ au lieu de 10 Mo pour un CD.

2. Codes singuliers

Une source est dite **stationnaire** quand la probabilité $P(S_t)$ ne dépend pas du temps. Cela veut dire que la probabilité $P(s_i)$ pour que le symbole s_i se trouve dans un message ne varie pas au cours du temps. La source est sans mémoire si la probabilité pour que le symbole s_i apparaisse ne dépend pas de ses apparitions précédentes.

On va ici s'intéresser aux sources discrètes. Un code source utilisable dans la pratique doit posséder un certain nombre de propriétés.

Certains codes sont **singuliers** ou **irréversibles**. Cela signifie que deux symboles source différents peuvent donner deux mots de code identiques. Par exemple si l'on a :

$$s_1 \mapsto 0 \quad s_2 \mapsto 1, \quad s_3 \mapsto 1 \text{ et } s_4 \mapsto 0$$

on ne sera pas capable de décoder le message 01 car cela pourrait être l'une des 4 séquences source : $s_1s_2, s_1s_3, s_4s_2, s_4s_3$. Un tel code est singulier ou irréversible. On ne peut pas le déchiffrer sans ambiguïté. C'est donc un codage à perte qui ne peut être utilisé que dans certaines applications où une perte d'information n'est pas cruciale comme pour coder la parole, des images ou des vidéos lorsque l'on doit les transmettre à faible débit.

Si, par exemple, dans une photo, on code deux niveaux de gris contigus de la même manière, cela ne rendra pas l'image inintelligible. De même, en musique, si l'on donne le même code à tous les signaux ayant une fréquence supérieure à 18 000 Hz, cela ne changera pas l'écoute du morceau.

Un code **non-singulier** est en revanche tel que

$$s_i \neq s_j \quad \Rightarrow \quad c_i \neq c_j$$

Deux messages source différents donnent deux messages codés différents. Si l'on veut transmettre un message alphanumérique, il est préférable d'avoir un code non-singulier.

3. Codes non-ambigus

Un code est **non-ambigu** quand chaque mot de code correspond à un et un seul message source. En fait il y a une bijection entre symboles source et les mots de code. À chaque symbole source correspond un mot de code et réciproquement.

$a \mapsto 1 \;\; b \mapsto 00$ et $c \mapsto 11$ est un exemple de code ambigu. En effet, si l'on reçoit le code 1111, on peut l'interpréter comme $aaaa, cc, aac, aca$ ou caa.

Un exemple de code non-ambigu est $a \mapsto 1 \;\; b \mapsto 00$ et $c \mapsto 10$. En effet, 10000 se décode comme abb et 1100 comme aab. Cependant, on est obligé d'avoir le message en entier pour pouvoir le décoder.

Le tableau 2 donne deux exemples de codes non-ambigus, c'est-à-dire à décodage unique, pour une source de 4 symboles et un codage binaire.

Code source	Code A	Code B
s_1	0	0
s_2	01	10
s_3	011	110
s_4	0111	111

Tableau 1

Ces codes sont à décodage unique et sans ambigüité. Par exemple, avec le code A, $00101 \Rightarrow s_1 s_2 s_2$. De même, avec le code B, $111010 \Rightarrow s_4 s_1 s_2$.

En revanche le code suivant (tableau 2) est ambigu. Le décodage n'est pas unique.

Code source	Code
s_1	0

s_2	10
s_3	11
s_4	110

Tableau 2

Par exemple, le code 110 peur être interprété comme $s_3 s_1$ ou s_4.

4. Codes sans préfixe

Parmi les codes non-ambigus, les **codes sans préfixe** sont intéressants. Si l'on considère une séquence c de longueur $n \geq 1$, on dit qu'elle est préfixe d'une autre séquence c' si les n premiers symboles de c' sont la séquence c. Par exemple, la séquence $abcd$ est un préfixe de la séquence *abcdefg*.

Le code suivant $a \mapsto 0 \quad b \mapsto 00$ et $c \mapsto 10$ n'est pas un code sans préfixe car $0\ (a)$ est le préfixe de $00\ (b)$.

Par contre le code $a \mapsto 0 \quad b \mapsto 10$ et $c \mapsto 11$ est un code sans préfixe.

5. Code instantané

Un **code instantané** est un code où chaque mot de code dans une chaîne de mots de code peut être décodé sans avoir à attendre le mot de code suivant et notamment la fin du message. Ceci permet d'accélérer le processus de décodage et nécessite moins de mémoire. En effet, le décodage peut se faire à la volée dès que l'on reçoit un mot sans attendre et stocker la suite.

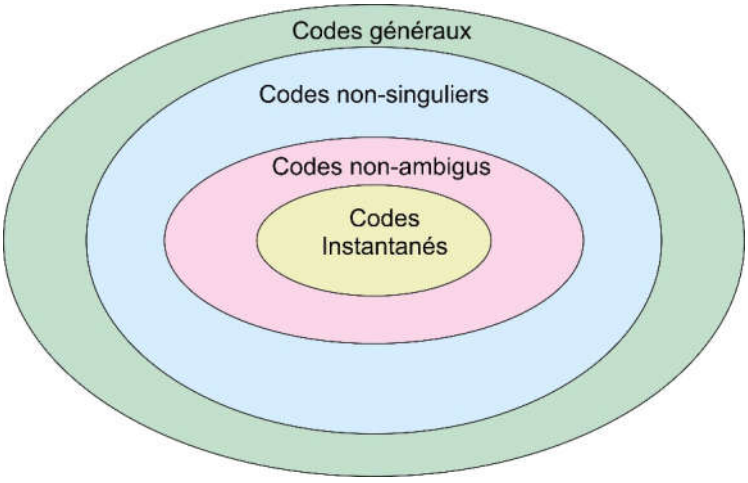

Figure 2

C'est dans le domaine des codes instantanés que les codes sans préfixe trouvent tout leur intérêt grâce aux deux propriétés importantes suivantes :

- Tout code sans préfixe est non-ambigu (mais l'inverse n'est pas vrai, il existe des codes non-ambigus qui sont avec préfixe)
- Un code est instantané si et seulement si il est sans préfixe

Il y a en fait toute une hiérarchie dans les codes qui est résumée dans la figure 2. Les codes instantanés étant par exemple contenus dans la famille des codes non-ambigus.

6. Code à longueur fixe

Dans un code à longueur fixe, tous les mots de code sont de la même longueur. C'est un codage facile à décoder mais peu efficace car il attribue aux symboles très fréquents la même longueur qu'aux symboles les plus rares.

Le tableau 3 donne un exemple.

Le code ASCII que nous verrons plus loin est un exemple de code à longueur fixe.

Code source	Code
s_1	00
s_2	01
s_3	10
s_4	11

Tableau 3

7. Code avec séparateur

On peut séparer chaque mot code par un séparateur. Pour cela on réserve un symbole. Le tableau 4 donne un exemple. Cela permet d'utiliser des mots de code de longueur variable.

Code source	Code
Séparateur	0
s_1	1
s_2	11
s_3	111
s_4	1111

Tableau 4

8. Le code de Morse

Le code Morse a été introduit bien avant que l'on parle de théorie de l'information. Il était initialement dédié aux transmissions militaires. Il a été largement utilisé pendant la guerre de sécession aux États-Unis et lors de la première guerre mondiale. Les mots de code sont constitués de points et de tirets. Lors d'une transmission télégraphique, un tiret dure trois fois plus longtemps qu'un point. L'espacement entre deux éléments d'une même

lettre est égal à 1 point et celui séparant deux lettres est égal à 3 points. L'espacement entre deux mots étant lui-même de 7 points. Le codage des lettres de l'alphabet ainsi que de certains signes et commandes est indiqué dans le tableau 5.

Une caractéristique du code Morse est que les mots de code représentant les lettres sont de longueur variable, les plus courts représentant les lettres les plus utilisées et les plus longs celles qui sont plus rares. Ainsi le « E » est représenté par un point seulement alors que le « Z » par ——••.

Le Morse a été le premier code à longueur variable utilisé. Bien qu'il ait été conçu avant que la notion de code optimal ait été introduite, il n'est pas loin de l'être alors qu'il a été conçu de façon empirique.

Quand on parle de Morse tout le monde pense au célèbre SOS (Save Our Souls) •••/———/•••.

A	•—	K	—•—	U	••—	4	••••—	;	—•—•—•
B	—•••	L	•—••	V	•••—	5	•••••	-	—•••—
C	—•—•	M	——	W	•——	6	—••••	/	—••—•
D	—••	N	—•	X	—••—	7	——•••	"	•—••—•
E	•	O	———	Y	—•——	8	———••	call T	—•—
F	••—•	P	•——•	Z	——••	9	—————•	error	•••••••
G	——•	Q	——•—	0	—————	.	•—•—•—	wait	•—•••
H	••••	R	•—•	1	•————	,	——••——	End M	•—•—•
I	••	S	•••	2	••———	?	••——••	End B	•••—•—
J	•———	T	—	3	•••——	:	———•••		

Tableau 5

9. Le code ASCII

Le code ASCII (American Standard Code Information Interchange) est un code de longueur fixe introduit dans les années soixante. Il est très utilisé aujourd'hui en informatique sous forme simple ou étendue. Sa particularité est d'être un code de longueur fixe. Chaque lettre alphabétique, chiffre ou caractère est représentée par 7 bits, ce qui permet en théorie de définir 128 codes différents. Le tableau 6 montre le code ASCII de quelques lettres majuscules ou minuscules, de chiffres et caractères spéciaux. Les valeurs ASCII entre 0 (NUL) et 31 correspondent à de caractères de contrôle. Dans le codage ASCII on ne fait pas la différence entre les caractères qui sont utilisés fréquemment et ceux qui ne le sont pas.

Décimal	Hexadécimal	Caractère	Décimal	Hexadécimal	Caractère	Décimal	Hexadécimal	Caractère
32	20	blank	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	'	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	DEL

Tableau 6

10. Codage source

Considérons une source discrète X utilisant un alphabet K. Cette source émet une suite de symboles x qui appartiennent à l'alphabet K. Chacun a la probabilité $p(x)$ d'apparaître. L'entropie de la source est :

$$H(X) = -\sum_X p(x)\log_2 p(x)$$

La valeur de cette entropie est bornée par $\log_2(K)$, où K est le nombre de symboles de l'alphabet :

$$H(X) \le \log_2(K)$$

L'égalité est atteinte lorsqu'il y a équiprobabilité des symboles de l'alphabet de la source.

Si la source émet régulièrement des symboles, toutes les τ par seconde, par exemple, elle émet $\frac{1}{\tau}$ symboles par seconde. Le **débit moyen d'information** peut alors être défini comme :

$$Q = \frac{H(X)}{\tau} \text{ bits/s}$$

Pour coder en binaire l'alphabet d'une source, il faut attribuer à chaque symbole un mot de code de L bits. Cela fait 2^L codes différents possibles pour une longueur L.

Pour qu'il soit possible de coder l'alphabet de taille K avec L bits, il faut que $2^L \ge K$. L'égalité a lieu lorsque K est un multiple de 2. Si ce n'est pas le cas il faudra choisir L tel que :

$$2^{L-1} < K < 2^L$$

Le nombre de bits de codage minimum pour coder un alphabet de K symboles est donc :

$$L = \text{int}\left[\log_2(K)\right] + 1$$

On doit aussi satisfaire aux deux relations suivantes :

$$\begin{cases} H(X) \le \log_2(K) \\ L \ge \log_2(K) \text{ car } (2^L \ge K) \end{cases}$$

Il y a égalité lorsque tous les symboles de la source sont équiprobables et si K est une puissance de 2.

L'efficacité du codage est d'autant plus grande que le nombre de codes inutilisés est petit. Par exemple, si l'alphabet contient 5 caractères A, B, C, D et E $(K = 5)$ il faut $L = 3$ bits pour le coder. En effet $2^2 = 4$ et $2^3 = 8$. On peut par exemple adopter le codage suivant :

A=000, B=001, C=010, D=011 et E=100. Les codes inutilisés sont 101, 110 et 111. Si l'alphabet source avait été A, B, C, D, E, F, G et H, les 8 codes auraient pu être utilisés.

Supposons, pour illustrer ce point, que $K = 26$ (lettres de l'alphabet). On a :

$$\log_2(K) = 4,7 \quad \text{et} \quad L = \text{int}\left[\log_2(K)\right] + 1 = 5$$

Si tous les symboles sont équiprobables, $p_i = \frac{1}{K}$ et :

$$H(X) = -\sum_{i=1}^{26} p_i \log_2 p_i = \log_2(K) \sum_{i=1}^{26} p_i = \log_2(K) = 4,7 \text{ bits}$$

On a alors :

$$\eta = \frac{H(X)}{L} = \frac{4,7}{5} = 94\%$$

Si nous prenons le cas où $K = 36$ (26 lettres de l'alphabet+ 10 chiffres), on a :

$\log_2(K) = 5,17$ et $L = 6$. Pour des symboles équiprobables $H(X) = 5,17$.

$$\eta = \frac{H(X)}{L} = \frac{5,17}{6} = 86\%$$

C'est moins bon que précédemment.

On définit aussi parfois la redondance $\mathcal{R}$ de la source comme :

$$\mathcal{R} = 1 - \eta = 1 - \frac{H(X)}{L}$$

On peut améliorer l'efficacité du codage en ne transmettant pas les symboles individuellement mais par blocs de J symboles. C'est ce que l'on appelle l'**extension de la source**. On passe de la source primaire constituée de N symboles à une source secondaire constituée de N^J symboles.

11. Arbres *n*-aires

Les arbres *n*-aires sont des objets mathématiques très utiles, notamment pour l'étude et le développement de codes instantanés.

La figure 3 montre un exemple d'arbre et quelques définitions.

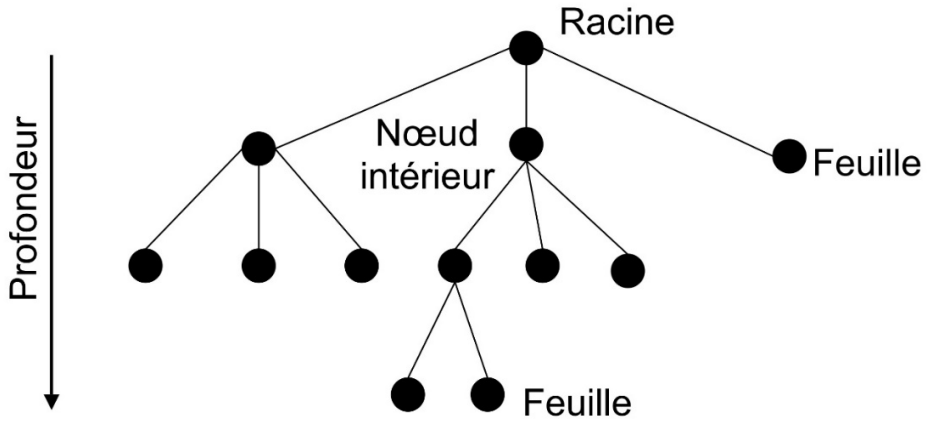

Figure 3

Un arbre est constitué de nœuds reliés entre eux. Il commence par une racine. Chaque nœud du graphe est soit un nœud intérieur, soit une feuille quand on est à une extrémité. On a les définitions suivantes relative à un graphe

- Un **nœud** possède au plus un prédécesseur : son père
- La **racine** est le seul nœud qui n'a pas de prédécesseur
- Un nœud qui n'a pas de successeur est une **feuille**
- Une **branche** est une suite finie de nœuds de la racine vers la feuille
- La **longueur d'une branche** est le nombre de nœuds moins 1
- La **profondeur d'un nœud** est égale à la longueur jusqu'à la racine.
- La **hauteur** d'un graphe est la longueur maximum d'une de ses branches
- L'**arité** est le nombre maximum de successeurs qu'un nœud peut avoir
- La **taille** est le nombre total de nœuds.

Voyons sur la figure 4 quelques caractéristiques d'un graphe. Le nœud 1 est la racine de l'arbre et c'est le père des nœuds 2, 3 et 4. Les nœuds internes sont les numéros 2, 3 et 9. Les feuilles sont les nœuds 5,6,7,8,10 et 4. Les nœuds 1, 3, 9 et 10 forment une branche de longueur 3. Les nœuds 1 et 4 forment une branche de longueur 1. La hauteur du graphe est de 3. La taille de l'arbre est de 10 et son arité de 3.

,

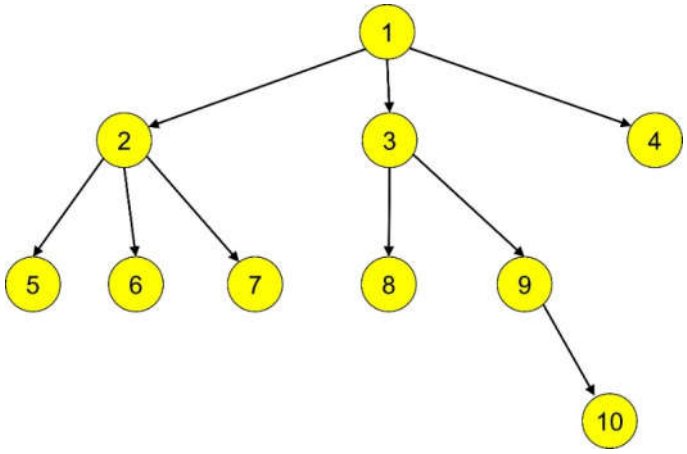

Figure 4

Les graphes permettent de construire et d'interpréter des codes instantanés. La figure 5 en montre un exemple.

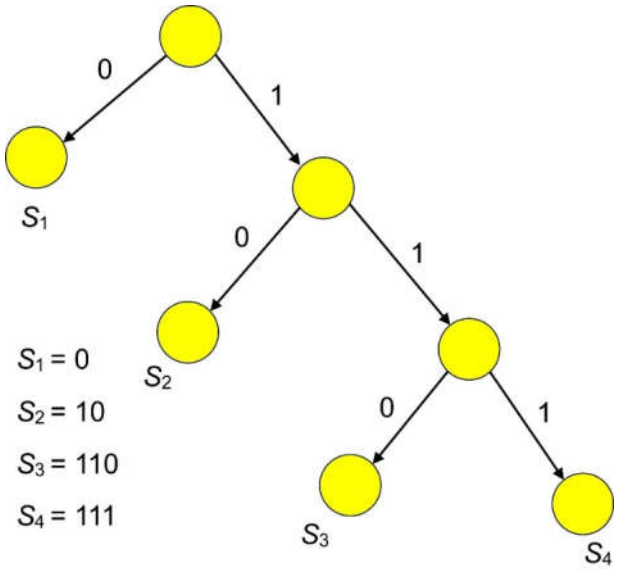

Figure 5

Ce sont les arbres binaires ($n = 2$) qui vont surtout nous intéresser dans le cas d'un traitement digital de l'information. Dans un arbre binaire, chaque nœud intérieur a une arité de 2. Il est dit complet lorsque toutes les feuilles ont la même profondeur. La figure 6 montre, dans partie gauche un arbre binaire incomplet et, dans la partie droite, un arbre binaire complet.

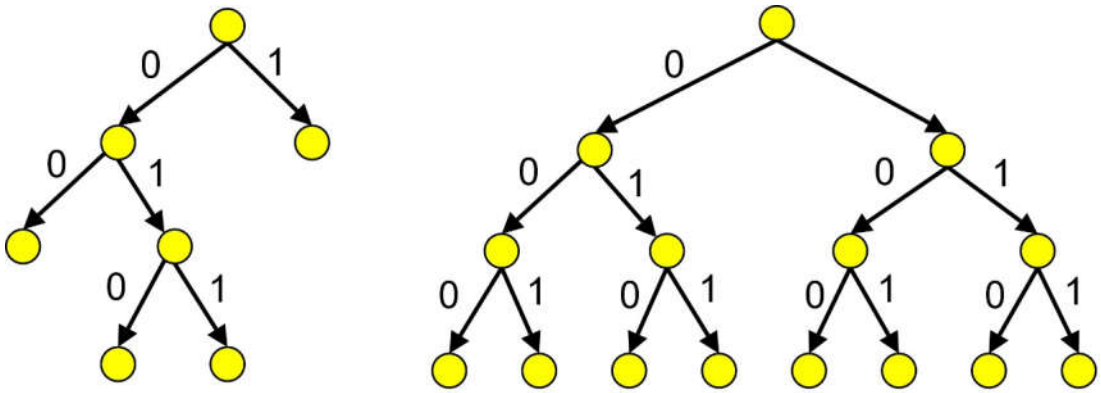

Figure 6

Un arbre de codage binaire est un arbre binaire dont les arcs sont étiquetés par les 2 symboles 0 et 1. Chaque symbole ne doit apparaître au plus qu'une fois à chaque nœud. L'intérêt des arbres de codage est qu'ils fournissent des codes instantanés.

L'arbre de code est une représentation commode pour écrire les algorithmes de codage et de décodage. Un code est dit **complet** lorsqu'il n'y a pas de feuille vide dans l'arbre de codage.

12. Inégalité de Kraft

L'inégalité de Kraft est une condition nécessaire et suffisante pour montrer l'existence d'un code instantané. Il s'agit d'un théorème dont l'énoncé est le suivant pour un code instantané binaire :

Il existe un code instantané binaire constitué de N *mots de code dont la longueur des mots de code sont des entiers positifs* $l_1, l_2, \ldots l_N$ *si et seulement si :*

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1$$

Lorsque l'égalité est réalisée, le code instantané est complet.

Ce théorème dit qu'un code instantané peut exister pour les longueurs de mot de code l_i. Par contre il ne dit pas si un code ayant des mots de code l_i est instantané. Cela peut d'ailleurs ne pas être le cas. Illustrons ceci sur deux exemples.

Prenons tout d'abord 3 symboles sources codé en binaire comme : $0, 10$ et 11. La somme $\sum_{i=1}^{N} 2^{-l_i} = 2^{-1} + 2^{-2} + 2^{-2} = 1$. Le code est ici instantané mais il faut le vérifier par ailleurs car la relation de Kraft ne suffit pas.

Prenons maintenant 3 mots de code de la même longueur que précédemment. $1, 00, 10$. Ce code satisfait à l'égalité de Kraft et il n'est pourtant pas instantané car 1 est préfixe du mot de code 10. On peut voir la différence entre les deux codes en les représentant avec un graphe (figure 7).

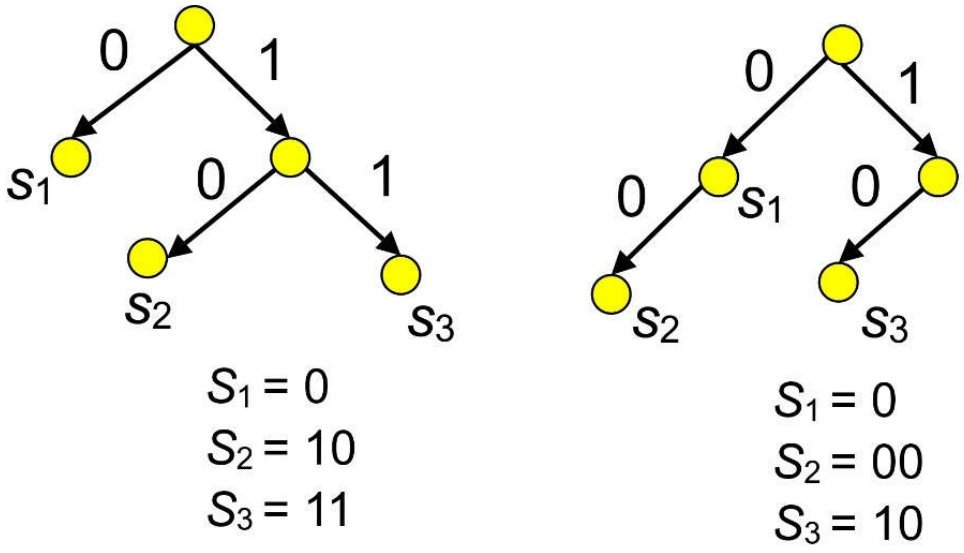

Figure 7

13. Entropie de la source et longueur des mots

Considérons une source composée de 4 symboles A, B, C et D apparaissant avec les probabilités indiquées dans le tableau 7. On va les représenter par trois codes binaires différents.

Symbole	Probabilité	Code 1	Code 2	Code 3
A	0,4	00	0	0
B	0,3	01	10	10
C	0,2	10	11	110
D	0,1	11	110	111
	H=1,85	*L*=2	*L*=1,7	*L=1,9*

Tableau 7

L'entropie de la source vaut $H(S)=\sum_i -p_i \ln_2 p_i = 1{,}85$ et la longueur moyenne des mots de code vaut, pour un code donné $L=\sum_i p_i l_i$. Le code 1 est de longueur fixe alors que les codes 2 et 3 sont de longueur variable, les mots de code les plus courts étant réservés aux symboles ayant la plus grande probabilité d'apparaître.

Si N est la taille de l'alphabet d'une source discrète X, la quantité moyenne d'information par symbole est l'entropie de la source $H(X)$. Celle-ci est maximale lorsque tous les symboles sont équiprobables et vaut dans ce cas $\ln_2 N$. Par conséquent on a toujours :

$$H(x) \le \ln_2 N$$

Si la source émet des symboles à une fréquence régulière, par exemple avec une fréquence υ, le débit moyen sera $Q = H(X) \times \upsilon$ bits/s

14. Codage par plage

Dans le codage par plages (run-length coding), on rassemble en paquets les signes s_i émis qui se suivent et qui sont identiques. Ceci est particulièrement intéressant si l'on veut transmettre un fax en noir et blanc. Il y a beaucoup de symboles blancs qui se suivent. En les regroupant, on peut réduire la taille du message à transmettre. C'est aussi intéressant pour les images.

Il n'y a pas de méthode générale pour guider le regroupement et ce type de de codage est plutôt un pré-encodage dans lequel on prépare le message de la source pour en réduire l'occupation avant de le coder avec d'autres méthodes.

Illustrons cela sur un exemple très simple. Supposons que l'on ait à coder la séquence suivante de 27 chiffres binaires :

$$101100100011011000000111101$$

Si l'on définit $s_1 = 0,\ s_2 = 1,\ s_3 = 11$ et $s_4 = 000$, le message devient alors :

$$s_1 s_2 s_3 s_1 s_1 s_2 s_4 s_3 s_1 s_3 s_4 s_4 s_3 s_3 s_1 s_2$$

On est passé à 16 symboles au lieu de 27.

Prenons comme autre exemple une ligne d'une image en noir et blanc dont le code est le suivant :

101000001000000010001010111110000010000000011

Il y a 45 caractères binaires représentant chacun un pixel. Le 0 pour un pixel blanc et 1 pour un pixel noir. Si on définit les mots de code suivants :

$b_i =$ i pixels blancs consécutifs

$n_j =$ j pixels noirs consécutifs

La ligne de code deviendra :

$$n_1 b_1 n_1 b_5 n_1 b_7 n_1 b_3 n_1 b_1 n_1 b_1 n_5 b_5 n_1 b_8 n_2$$

Ce qui est plus court que le message initial.

15. Codage de Shannon-Fano

Le codage de Shannon-Fano a pour but de maximiser l'entropie à la sortie du codeur. Son principe est de séparer de manière successive le message source en groupes tels que la somme des probabilités des messages de chaque groupe soient aussi voisines que possible.

Le processus est le suivant :

- Classer les différents messages de la source dans l'ordre de probabilités décroissantes.

- Diviser l'ensemble des messages en deux sous-ensembles de telle manière que leur probabilité cumulée soit à peu près égale.
- Attribuer le chiffre 1 au premier sous-ensemble et 0 au second (ou l'inverse).
- Répéter le processus sur tous les sous-ensembles et s'arrêter lorsque les sous-ensembles n'ont plus qu'un élément.

Prenons un exemple pour illustrer la procédure. Soit une source de 7 symboles avec les probabilités indiquées dans le tableau 8, déjà classées par ordre décroissant.

Symbole	A	B	C	D	E	F	G
Probabilité	0,4	0,2	0,15	0,1	0,05	0,05	0,05

Tableau 8

L'entropie de la source est :

L'entropie de la source est :

$$H = -\left[0,4\times\log_2 0,4 + 0,2\times\log_2 0,2 + 0,15\times\log_2 0,15\right]$$
$$-\left[0,1\times\log_2 0,1 + 3\times 0,05\times\log_2 0,05\right] = 2,384 \text{ bits}$$

Comme il y a $N = 7$ valeurs pour la source, l'entropie maximale est $\log_2 7 = 2,81$ bits.

La redondance de la source est $\mathcal{R} = 1 - \dfrac{2,384}{2,81} = 0,15$, soit 15%.

Pour un code binaire de longueur fixe n, il faut que $2^n \geq 7$, soit $n = 3$.

La figure 8 montre les codages de Shannon-Fano que l'on peut obtenir.

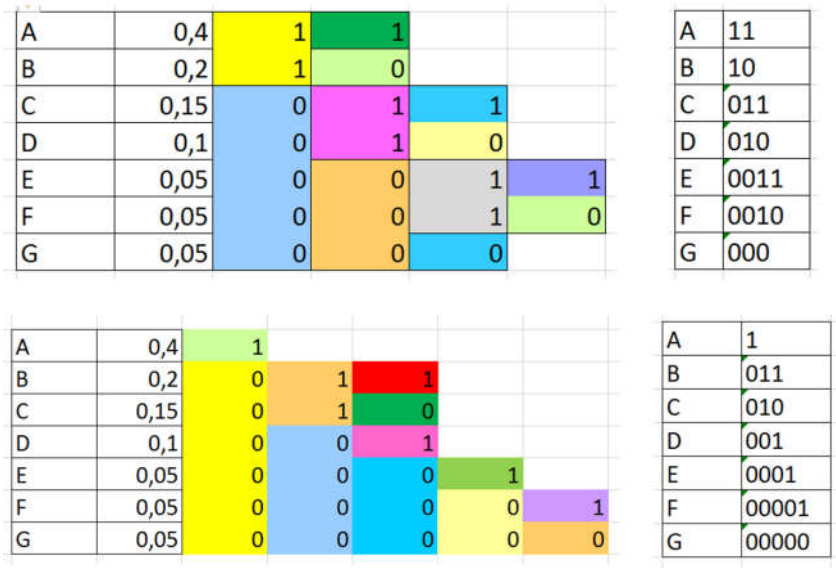

Figure 8

On voit sur cet exemple que l'on peut avoir deux codages de Shannon-Fano différents. Dans le premier cas on a une longueur moyenne de 2,5 bits et dans le second 2,45 bits.

On aurait pu partir du 0 au lieu de 1 pour les symboles les plus probables comme on le voit dans l'exemple du tableau 9 ci-dessous :

Symbole	Probabilité	Cycle 1	Cycle 2	Cycle 3	Code
s_1	0,5	0			0
s_2	0,2	1	0		10
s_3	0,2	1	1	0	110
s_4	0,1	1	1	1	111

Tableau 9

16. Codage de Huffmann

Le principe du codage de Huffmann (1952) est de coder les caractères de l'alphabet dont la probabilité est la plus faible avec des mots longs et ceux dont la probabilité est la plus grande avec des mots courts. Ces derniers seront, pour le cas qui nous intéresse, du binaire, donc une succession de 0 et de 1.

Le codage de Huffmann permet d'envoyer des messages plus comprimés que si l'on associe chaque caractère initial à un mot binaire de longueur fixe.

Dans la pratique, on procède selon les étapes suivantes :

- On classe les différents caractères de la source dans l'ordre de probabilités d'apparition croissante.
- On regroupe les deux caractères de plus faible probabilité en un seul dont la probabilité est la somme des probabilités de chacun d'entre eux.
- On attribue un chiffre, 0 ou 1 selon la branche du graphe
- On reclasse les probabilités par valeurs croissantes
- On refait les opérations décrites ci-dessus jusqu'à ce que l'on épuise tous les caractères.

Voyons cela sur un exemple concret. On veut transmettre le message « ETEPERETENETETENETE ». C'est un message inintelligible car il a été crypté. Appliquons le codage de Huffmann. Dans ce message le E apparaît 10 fois, le T 5 fois, le N 2 fois et P et R une seule fois (tableau 10). Dans la figure 9, on classe les lettres selon leur probabilité croissante (haut de la figure) et on commence par regrouper le P et le R. Une fois cela fait on regroupe le P et le R (deuxième ligne)

Caractère	E	T	N	P	R
Fréquence	10	5	2	1	1

Tableau 10

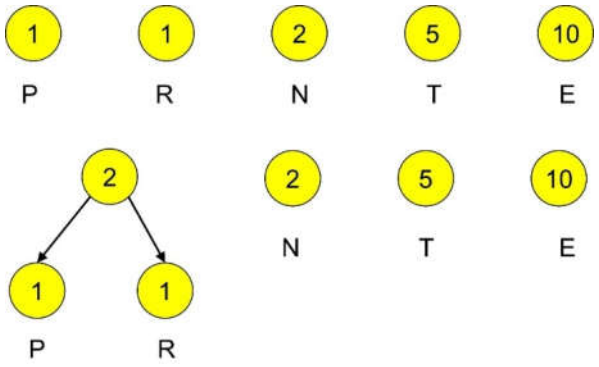

Figure 9

Dans les figures suivantes (10, 11 et 12), le processus de codage est développé.

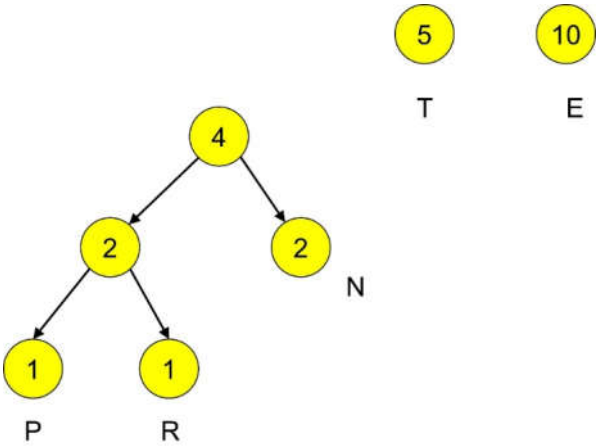

Figure 10

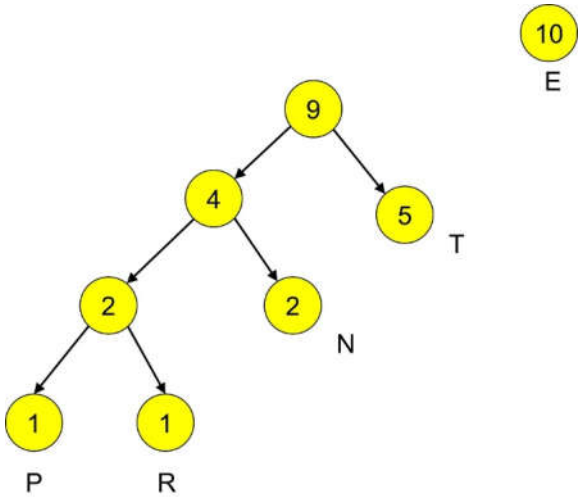

Figure 11

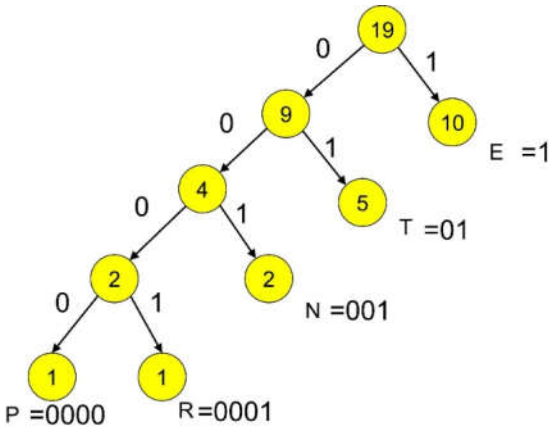

Figure 12

On aboutit au codage du tableau 11 :

E	T	N	P	R
1	01	001	0001	0000

Tableau 11

La longueur du message codé est de 10 (E)+10 (T) +6 (N) +4 (R) + 4 (P), soit 34 bits pour coder le message. Si l'on avait utilisé un code à longueur fixe, comme l'ascii par exemple, il aurait fallu 19×8=152 bits. On a divisé par plus de 4 la longueur du message.