Codes détecteurs correcteurs

Arnaud Labourel
Courriel : arnaud.labourel@lif.univ-mrs.fr

Université de Provence

15 novembre 2011

Coder et transmettre

Codage de l'information

- Information codée par des 0 et des 1
- Codage transmis sous la forme de courants, ondes, etc.

Erreurs de transmission

- Remplacements de 0 par 1 (et inversement)
- Un bit par microseconde : accumulation d'erreurs de transmission

  taux d'erreur de 10^{-6} et connexion à 1Mo/s, en moyenne 8 bits erronés transmis chaque seconde !

Exemples

Exemples concrets

Internet Protocole TCP : erreur détectée, correction par retransmission

Le CD Rayures ou impuretés du support (peu fréquentes mais très volumineuses) : correction à la volée

Le port série Correction de petites erreurs relativement fréquentes mais isolées : correction immédiate

Contraintes diverses

- Faible coût d'implantation
- Pas d'altération de la vitesse de transmission
- Fiabilité !

Quelles solutions ?

Agir sur le canal de transmission

Rendre le taux d'erreur négligeable

- Impossibilité de remplacer toutes les infra-structures existantes !
- Taux nul impossible

Agir sur le message transmis

- Etre capable de détecter si des erreurs ont eu lieu
- Etre capable de les corriger au mieux

Principe des codes correcteurs d'erreurs

Détecter et corriger les erreurs

Détecter une erreur : et après ?

- Recommencer la transmission ?
  - Pas toujours possible
  - Manque de temps
  - Nouvelles erreurs possibles
- Corriger l'erreur

Théorie des codes

Comment coder les messages pour favoriser détection et correction des erreurs de transmission

00111010101100011111000011111101101100011111001011000111100
00011100011111010001110001111111111110001110000011111111000000

(01011010111101001100)

Principe de la redondance

Pour me transmettre un numéro de téléphone, parmi les solutions possibles :

Message transmis	Message reçu
0654122235	0654172236
`zero six cinquante-quatre douze deux deux trente-cinq`	`zer0 slx cinquamte-quatte doyze deux deyx trsnte-c1nq`

- Information concise : difficile à corriger
- Information redondante : détection et correction d'erreur possible avec contexte !

Exemples de redondances

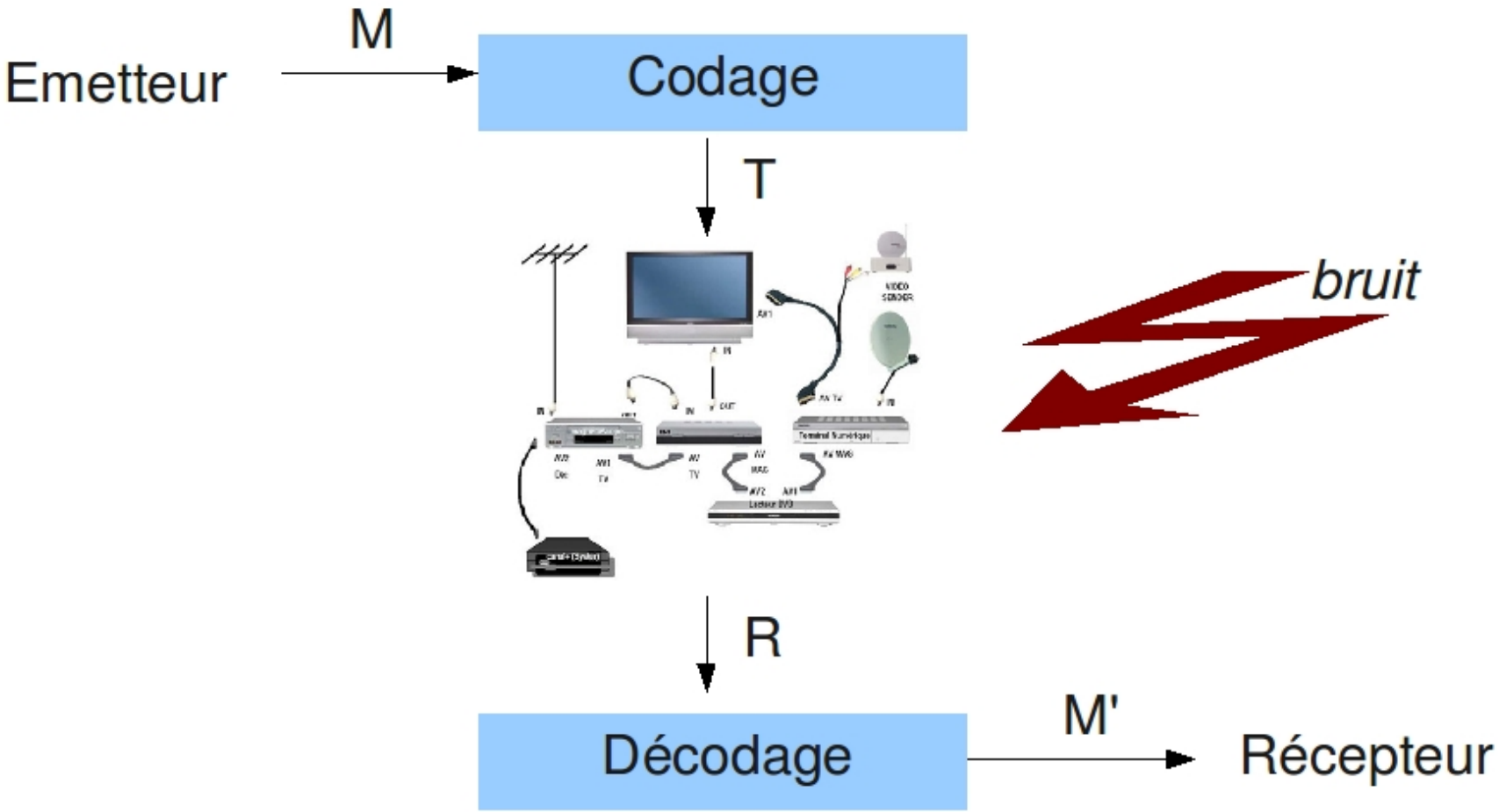
Le plus souvent, répétition du message !

- Au téléphone :
  - Répétition des numéros
  - Epeler le mot (A comme Abracadabra, R comme Ribambelle, etc.)
- Alphabet radio universel (Alpha Bravo Charlie Delta etc.)
- Orthographe traditionnelle moins sensible aux erreurs que langage type SMS
- Code génétique

Utilité du codage

- Détection et correction d'erreurs : redondance
- Compression (abréviations, code de Huffman, codec, archivage, etc.)
- Changement d'alphabets (code Morse, ASCII, etc.)
- Cryptographie (rendre le décodage difficile)

Schéma général de la communication



Formalisation du codage

Cas général

Soient les alphabets A^n et B^m.
Un codage est une application injective $\phi : A^n \to B^m$

Nécessité de l'injection : décodage

Soit $M' = \phi(M)$.
Décodage : à partir de M', retrouver M tel que $\phi(M) = M'$
Si ce M existe, il doit être unique !

Cas du codage binaire

$A = B = \mathbb{B}$, et $m > n$.

Addition modulo 2

Codage des messages = codage binaire

- Un message de n bits = un mot binaire de longueur n
- Opérations classique de $\mathbb{B}^n$

Un nouvel opérateur dans $\mathbb{B}$ (XOR)

$$0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1 \quad 1 \oplus 1 = 0$$

Addition modulo 2 (cont'd)

Propriétés

$$a \oplus b = b \oplus a \tag{1}$$

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c \tag{2}$$

$$a \oplus 0 = a \tag{3}$$

$$a \oplus a = 0 \tag{4}$$

De plus :

$$a \oplus b = c \iff a \oplus c = b \iff b \oplus c = a \tag{5}$$

Addition modulo 2 (cont'd)

Extension à $\mathbb{B}^n$

Si $a = a_1 a_2 \dots a_n$ et $b = b_1 b_2 \dots b_n$ sont deux mots binaires, leur somme modulo 2 est le mot binaire ayant pour $p^{\text{ème}}$ bit $a_p \oplus b_p$.

a	01001110
b	11011100
$a \oplus b$	10010010

(on vérifie dans $\mathbb{B}^n$ les propriétés 1 à 5 en remplaçant 0 par le mot binaire de taille n dont tous les bits sont nuls)

Poids $w(a)$ d'un mot binaire a

Nombre de bits dans a égaux à 1 : $w(0110010000110001) = 6$

$a \oplus b$ indique une distance

Interprétation de $a \oplus b$

- a et b mots binaires de longueur n
- $p^{\text{ème}}$ bit de $a \oplus b$: 0 si $a_p = b_p$ et 1 si $a_p \neq b_p$
- $a \oplus b$ indique les endroits où a et b diffèrent
- $w(a \oplus b)$ indique le nombre de différences

Définition (Distance de Hamming)

La distance de Hamming entre deux mots binaires a et b de taille n est le poids de $a \oplus b$.

$$d(a, b) = w(a \oplus b)$$

Propriétés de la distance de Hamming

Exemple

$$d(111001111000, 101000111001) = 3$$

$$d(a,b) = d(b,a) \quad \text{(symétrie)} \tag{6}$$

$$d(a,b) \geq 0 \quad \text{(positivité)} \tag{7}$$

$$d(a,b) = 0 \quad \text{si et seulement si } a = b \tag{8}$$

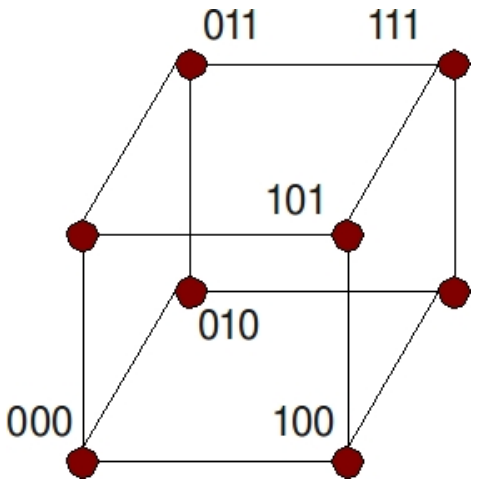
$$d(a,b) + d(b,c) \geq d(a,c) \quad \text{(inégalité triangulaire)} \tag{9}$$

$$d(a \oplus x, b \oplus x) = d(a,b) \quad \text{(invariance par translation)} \tag{10}$$

Interprétation géométrique

Algèbre de Boole sur les mots binaires !

Eléments de $\mathbb{B}^n$ disposés dans un n-cube :



Théorème

La distance $d(a, b)$ est le nombre d'arêtes qu'il faut longer pour se rendre de a à b par le chemin le plus court possible.

Vecteur d'erreur

Message envoyé et message reçu

- Expéditeur : message T de longueur n
- Destinataire : message R de longueur $p = n$
- Tout va bien si $R = T$, erreur de transmission si $R \neq T$

Définition (Vecteur d'erreurs)

- *Le vecteur d'erreur entre un message transmis T et un message reçu R est le vecteur $e = T \oplus R$*
- $w(e) = d(T, R)$ *est le nombre de bits mal transmis.*
- $T = e \oplus R$ *et* $R = T \oplus e$
- *Exemple :* $e = 1011 \oplus 1101 = 0110$

Code et mots de code

Détecter une erreur de transmission ?

- Si destinataire connaissait e, il pourrait retrouver T !
- Mais e reste inconnu !
- Accord entre expéditeur et destinataire sur certaines particularités de messages

  Définition du code $\mathcal{C}$: ses éléments sont des mots de code
- Le code favorise la détection d'erreurs, dans le cas où $R \notin \mathcal{C}$
- Erreurs non reconnues si $R \in \mathcal{C}$ pourtant erroné
- Exemple : triplement de chaque bit

$$T = 000111111111, R = 000111111000 \in \mathcal{C}$$

Corriger une erreur de transmission ?

Ensemble des vecteurs d'erreur possibles pour R

- Le destinataire connaît $\mathcal{C}$ et R !
- Il peut déterminer $\Gamma(R) = \{C \oplus R | C \in \mathcal{C}\}$: $e \in \Gamma(R)$
- Exemple : $R = 000110, \mathcal{C} = \{\text{m.b}\}$. où chaque bit est triplé

$$\mathcal{C} = \{000000, 000111, 111000, 111111\}, \Gamma(R) = \{000110, 000001, 111110, 111001\}$$

Une approche probabiliste pour corriger

- Le destinataire souhaite corriger R si une erreur est suspectée
- Il sait que $e \in \Gamma(R)$, mais ne peut le connaître *avec certitude*
- *Paris de corrections avec évaluation des risques*

Estimation des risques ?

Canal binaire

- Dispositif pour la transmission de bits
- Risque d'erreur d'un canal binaire estimé par

$$p = \frac{\text{nombre de bits faux}}{\text{nombre de bits transmis}}$$

- Expédition d'un bit = phénomène aléatoire à deux issues
  1. Le bit est *bien* transmis
  2. Le bit est *mal* transmis
- Probabilité d'erreur du canal = p
- $q = 1 - p$
- p très petit (sauf si canal très mauvais !)

Quelques hypothèses sur le canal

Canal symétrique

- p identique, que l'on transmette des 0 ou des 1
- Pas toujours valable (présence ou absence de trous sur un CD...)

Canal sans mémoire

- Les erreurs ne se groupent pas et même répartition des erreurs
- *Transmission de chaque bit indépendante des autres*
- Pas toujours valable non plus !

Transmission d'un message : un phénomène aléatoire

- Expédition d'un mot binaire de longueur n = répétition n fois de l'expédition d'un bit : bien ou mal transmis
- Issues du phénomène aléatoire E = vecteurs d'erreur
- Nombre d'erreurs E suit une loi binomiale $B(n, p)$

Rappels

Soit Φ un ph. aléa. ne comprenant que deux issues, X et Y, muni d'une loi p. Soit $x = p(X)$ et $y = p(Y)$, avec $y = 1 - x$.
Si n essais répétés sont indépendants, alors :

- $p(M) = x^k y^{n-k}$ si M est un mot particulier (cible connue) contenant k issues X et $n - k$ issues Y
- $p(M) = C_n^k x^k y^{n-k}$ si M est un mot quelconque contenant k issues X et $n - k$ issues Y.

En pratique...

Soit un canal sur lequel le taux d'erreurs $p = 10^{-2}$ (et $q = 0.99$).
Envoi de messages de 64 bits sur ce canal

Probabilité d'avoir :

- aucune erreur de transmission :

$$P(E = 0) = (1 - p)^n = q^n = 0.99^{64} \sim 0,526$$

- une seule erreur : $P(E = 1) = np(1 - p)^{n-1} \sim 0,340$
- deux erreurs : $P(E = 2) = \frac{n(n-1)}{2} p^2 (1 - p)^{n-2} \sim 0,108$
- trois erreurs : $P(E = 3) = C_n^3 p^3 (1 - p)^{n-3} \sim 0,023$
- plus de 4 erreurs : $P(E > 4) < 0,0005$

Résultat fondamental

Théorème

Lorsque l'on transmet des mots binaires de longueur n sur un canal binaire symétrique et sans mémoire, dont la prob. d'erreur est p :

1. *Si l'on se donne e, la probabilité que le vecteur d'erreur soit e est* $p^{w(e)}q^{n-w(e)}$
2. *Si l'on se donne k, la probabilité que le nombre d'erreurs de transmission soit k est :* $C_n^k p^k q^{n-k}$
3. *Si l'on transmet T, la probabilité de recevoir R est* $p^{d(T,R)}q^{n-d(T,R)}$

Exemple

On transmet $T = 0110$, soit $p = 0.001$: visualisons $p(R)$ et le nombre d'erreurs d

Probabilités de R, $T = 0110, p = 0.001$

R	d	$P(R)$	valeur ($p = 0.001$)
0000	2	p^2q^2	$0.0000009\ldots$
0001	3	p^3q^1	$0.0000000009\ldots$
0010	1	p^1q^3	$0.0009\ldots$
0011	2	p^2q^2	$0.0000009\ldots$
0100	1	p^1q^3	$0.0009\ldots$
0101	2	p^2q^2	$0.0000009\ldots$
0110	0	p^0q^4	$0.9\ldots$
0111	1	p^1q^3	$0.0009\ldots$
1000	3	p^3q^1	$0.0000000009\ldots$
1001	4	p^4q^0	$0.000000000001\ldots$
1010	2	p^2q^2	$0.0000009\ldots$
1011	3	p^3q^1	$0.0000000009\ldots$
1100	2	p^2q^2	$0.0000009\ldots$
1101	3	p^3q^1	$0.0000000009\ldots$
1110	1	p^1q^3	$0.0009\ldots$
1111	2	p^2q^2	$0.0000009\ldots$

Comment détecter et corriger ?

Interprétation

- $p \ll q$: probabilité de substitution de M_1 par M_2 diminue très vite quand $d(M_1, M_2)$ augmente
- Plus le poids de e augmente, moins il est probable
- Idée : e est l'élément de $\Gamma(R)$ de plus faible poids
- Remplacer le mot reçu par le mot vraisemblable le plus proche

Stratégie : un pari

1. e est le mot de plus petit poids dans $\Gamma(R)$
2. Remplacement de R par $T = e \oplus R$ ($e = 0$ si $T = R$)

Correction la plus vraisemblable, mais pas toujours valable

Exemple : mots $0xx\cdots x0$

- Soit $\mathcal{C} = \{0000, 0010, 0100, 0110\}$
- Réception de $R = 0111$, $R \notin \mathcal{C}$
- Calcul de $\Gamma(R)$:

$$\Gamma(R) = \{0000 \oplus 0111, 0010 \oplus 0111, 0100 \oplus 0111, 0110 \oplus 0111\}$$

$$\Gamma(R) = \{0111, 0101, 0011, 0001\}$$

e	$w(e)$	$P(e)$	T
0111	3	p^3q	0000
0101	2	p^2q^2	0010
0011	2	p^2q^2	0100
0001	1	pq^3	0110

- Choix de e de plus petit poids

Codage par blocs : très répandu

L'expéditeur :

1. Découper le message binaire en blocs (paquets de bits) de longueur fixe
2. Coder chaque bloc avec ajout de *bits de contrôle* : mots de code
3. Envoi des mots de code

Le receveur

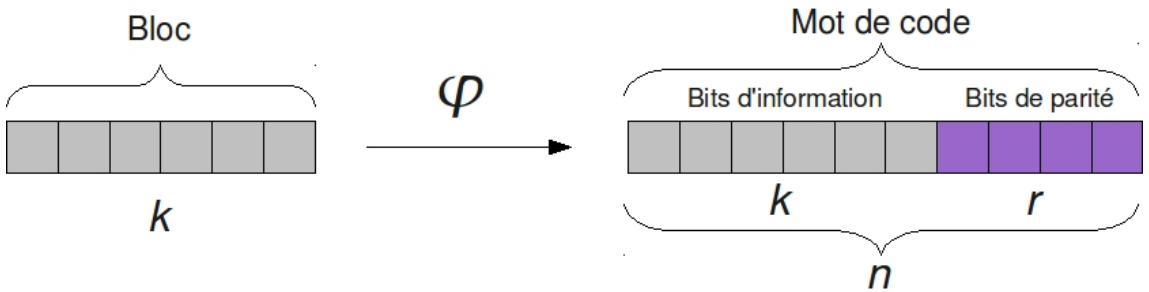
1. Réception des mots de code et détection d'erreur (le mot est-il un mot de code ?)
2. Correction éventuelle du mot reçu en le remplaçant par le mot de code le plus proche

Constitution du mot de code

Codage des blocs

- Bloc de dimension k = un mot binaire de longueur k
- Ajout de r bits de contrôle (ou bits de parité)
- Obtention de mots binaires de longueur $n = k + r$
- Mots de code de taille $n \subseteq$ mots binaires de taille n

Codage systématique



Codage = application

Quelle application ?

- Ensemble des blocs $= \mathbb{B}^k$, ensemble des messages $= \mathbb{B}^n$
- 2^k mots de codes et 2^n messages
- Codage $= \phi : \mathbb{B}^k \rightarrow \mathbb{B}^n$ associe un mot de code à un bloc

Rendement (taux) d'un code

Proportion du bits d'information parmi les transmis $\tau = \frac{k}{n}$

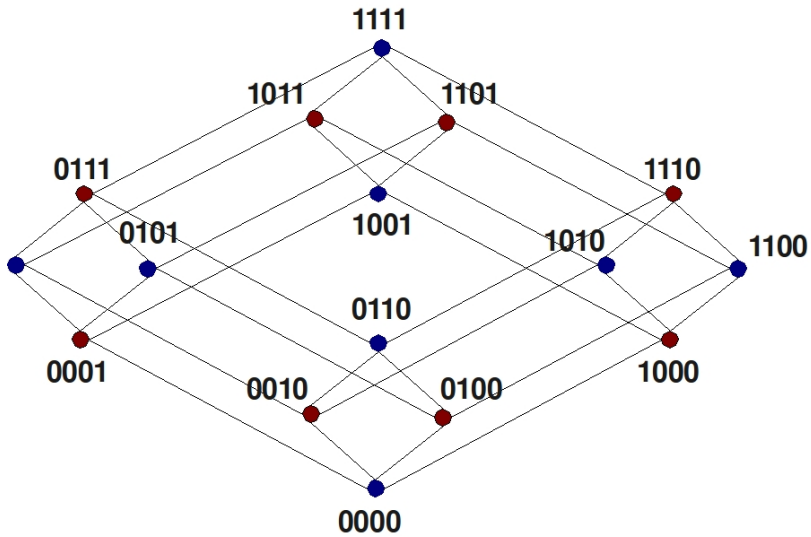
- Le plus élevé possible
- Métaphore du colis : résistance (poids) et prix de l'emballage
- Compromis sécurité *vs.* coût de l'envoi
- proportion de mots de code $= 2^k/2^n = 2^k/2^{k+r} = 1/2^r$

Test de parité

Principe

Un bit de contrôle pour garantir la parité du nombre de 1 ($r = 1$)
Dans le cas $k = 3$: $n = 4$ et $\tau = 3/4$
Exemple :$\Phi(001) = 0011, \Phi(110) = 1100$

Mots de codes



Test de parité (cont'd)

Exemples de (non-)détection

- Réception de $R = 1011$: une ou trois erreurs commises
- Réception de $R = 1111$: aucune, 2 ou 4 erreurs commises
  - Semble correct : on laisse passer !

Correction impossible

Si erreur détectée, le bit de contrôle ne fournit aucune information sur l'endroit de l'erreur

- $\mathcal{C} = \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$
- $R = 1011 \notin \mathcal{C}$ une ou trois erreurs ?
- 4 vecteurs d'erreur de poids $1 = \{1000, 0010, 0001, 0100\}$
- T les plus vraisemblables : $T \in \{0011, 1001, 1010, 1111\}$

Test de parité (cont'd)

Risque pris lors de l'acceptation d'un mot de code

- $p(R = T) = q^4$,
- $p(R \neq T) = 1 - q^4 = 1 - (1-p)^4 = 4p - 6p^2 + 4p^3 - p^4$
- Cas d'erreur de jugement : si 2 ou 4 erreurs
  - $p(\text{deux erreurs}) = 6p^2q^2$,
  - $p(\text{quatre erreurs}) = p^4$
  - donc $p(\text{laisser passer un message faux}) = 6p^2q^2 + p^4$
- Proportion de messages faux non détectés (FP) :

$$\frac{6p^2 - 12p^3 + 7p^4}{4p - 6p^2 + 4p^3 - p^4} \sim \frac{6p^2}{4p} \sim 1.5p \text{ si } p \text{ petit}$$

- Si $p = 0.001$: $0,15\%$ de FP (*vs.* 100% sans codage !)

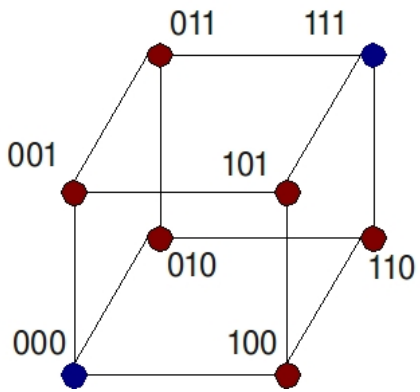
Codage par répétition

Principe

Bits d'informations transmis un à un, mais en les triplant.

- $\Phi(1) = 111, \Phi(0) = 000, \Phi(01) = 000111$
- Avec $k = 1, r = 2, n = 3$, pauvre rendement $\tau = 1/3$

Taux de détection (cas de deux mots de codes)



Si $T = 000$ et $R = 111$, l'erreur n'est pas détectée !
$p(\text{faux non détectés}) = \frac{p^3}{3p-3p^2+p^3} \sim p^2/3$ quand p est petit.

Codage par répétition (cont'd)

Capacité de correction

3 bits faux ils sont acceptés, pas de correction

2 bits faux majoritaires, donc le troisième est corrigé

1 bit faux minoritaire, il est corrigé

pas de bit faux pas de correction.

$$p(\text{mauvaise correction}) = p^3 + 3p^2q = 3p^2 - 2p^3$$

Proportion de messages faux qui le restent après correction

$$\frac{3p^2 - 2p^3}{3p - 3p^2 + p^3} \sim p$$

Récapitulatif

Pari que l'erreur commise est l'erreur la plus probable

Méthode pratique pour corriger un message

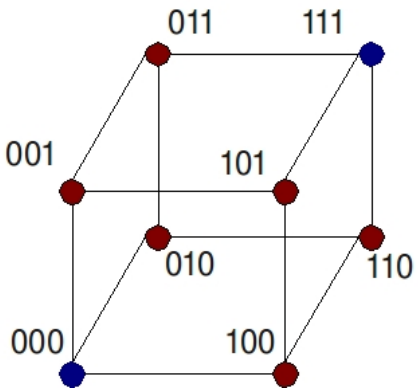
1. Déterminer $\Gamma(R)$ (vecteurs d'erreurs pour R)
2. Déterminer le message de plus faible poids dans $\Gamma(R)$
3. Ajout modulo 2 ($\oplus$) de ce message à R : obtention du mot de code le plus vraisemblable qui remplacera R

Sur un n-cube

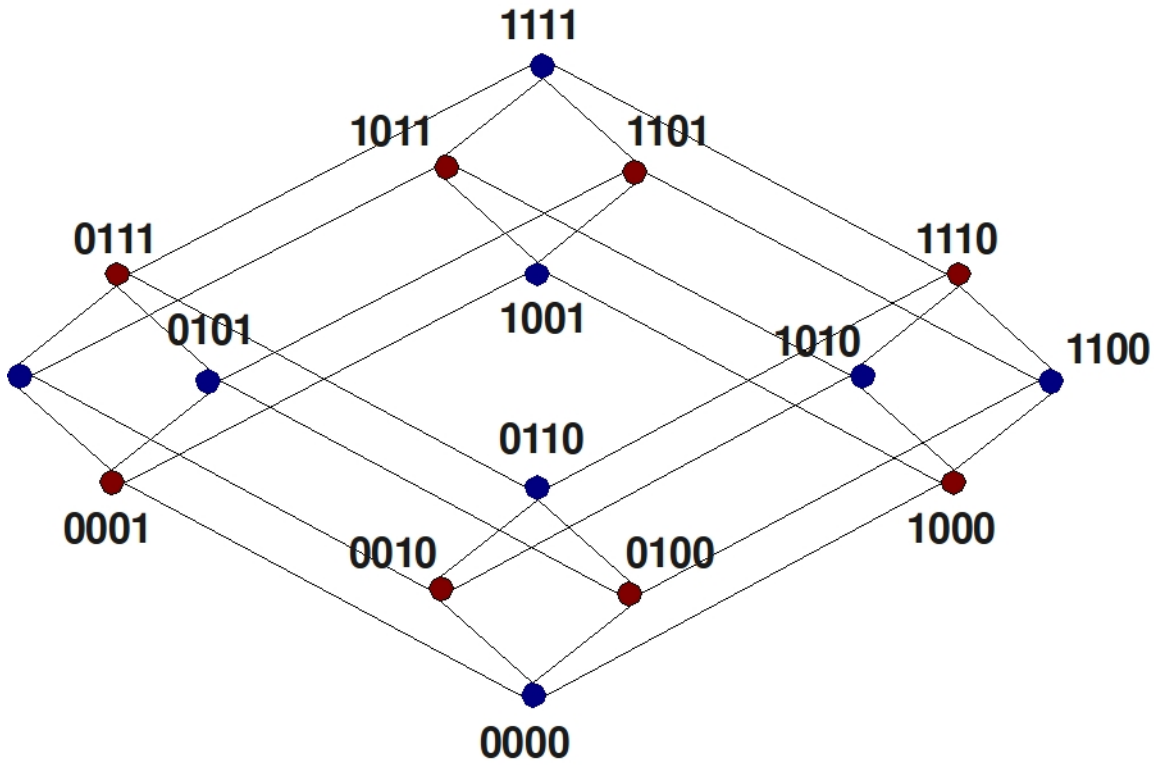
- Repérer le message $M \in \mathcal{C}$ le plus proche de R
- Quid en cas d'ex-aequo ?

Visualisation des corrections concurrentes

Répétition



Parité



Distance minimale

Définition (Distance minimale)

La plus petite distance séparant deux mots de code distincts est appelée la distance minimale *du code, notée* d.

Exemples

- Test de parité : $d = 2$
- Codage par répétition : $d = 3$

Théorème

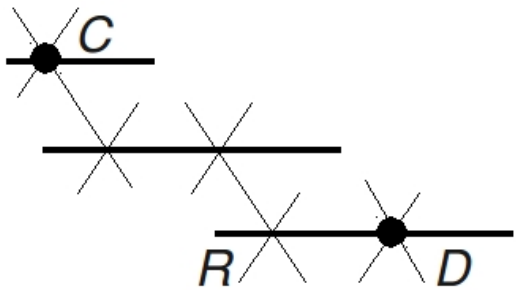
Le receveur détecte de façon certaine TOUS les messages faux, tant que le nombre d'erreurs N *vérifie* $0 < N < d$.
Certains messages faux comportant d *erreurs (ou plus) ne sont pas détectés.*

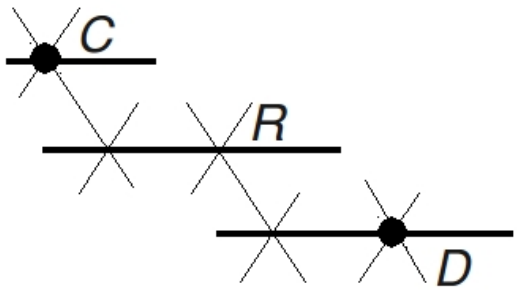
Distance minimale (cont'd)

Théorème

Les messages sont bien corrigés tant que le nombre d'erreurs N vérifie $0 \leq N < d/2$ (strict).
Les messages faux tels que $d/2 \leq N$ ne sont pas toujours bien corrigés.

Exemple





Nombre d'erreurs corrigées

$$t = \lfloor d/2 \rfloor = \frac{d}{2} - \frac{1-(-1)^d}{4}$$

d	t
1	0
2	0
3	1
4	1
5	2
6	2

D'après le théorème précédent :

- Tout message faux ayant t erreurs ou moins est corrigé de façon parfaite
- Il existe des messages faux mal corrigés qui ont $t+1$ erreurs
- La correction peut parfois être bonne même si $N > t$

Triplet de description d'un code $[n, k, d]$

Exemples

- Test de parité : code $[4, 3, 2]$
- Codage par répétition : $[3, 1, 3]$

Intérêt d'une grande d

Pour plus de détection et de correction d'erreurs

- Mots de code éloignés
- Si k est fixé : nombre de mots de code fixé
- Si k fixé : intercaler le plus possible de messages entre mots de code : augmenter n, mais alors k/n diminue !
- Bon code : d et n grands

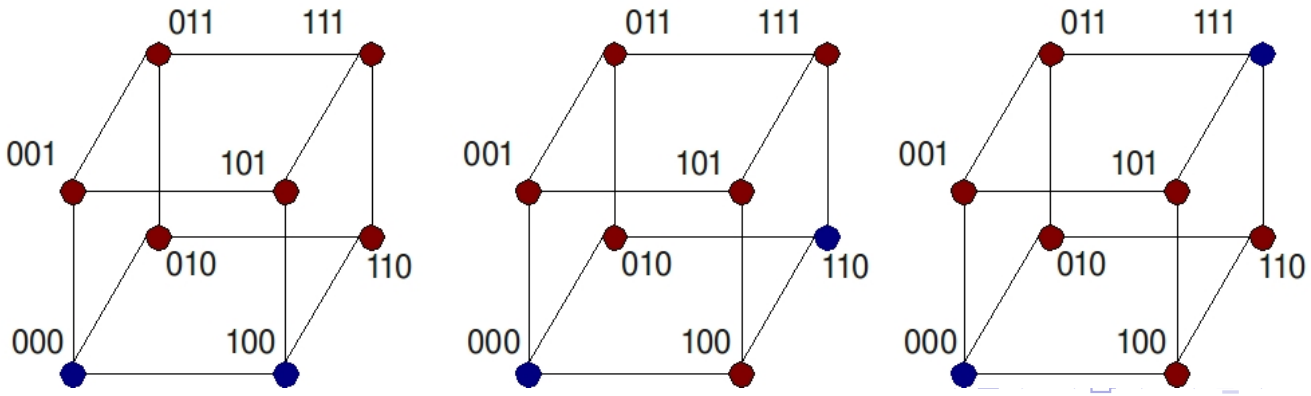
Interprétation sur les n-cubes

k et r fixés. Donc $\mathbb{B}^n$ fixé : tous les messages possibles, mots de code ou pas.

A résoudre

Comment marquer d'un point bleu 2^k sommets d'un n-cube de telle sorte que tous les sommets bleus soient le plus éloignés possible les uns des autres ?

Exemples : $k = 1,\ r = 2,\ d \in \{1, 2, 3\}$



Inégalité de Hamming

Objectif, n fixé

- Majorer t, donc d
- On a toujours $d \leq n$, mais grossier !

Inégalité de Hamming

Les quantités n, r, et t, sont liées par l'inégalité de Hamming :

$$C_n^0 + C_n^1 + C_n^2 + \cdots + C_n^t \leq 2^r$$

Soit message M de taille n, et $p \leq n$, il y a exactement C_n^p messages situés à la distance p de M (modification de p bits parmi n) ; Considérons la sphère $S(M, m)$ des messages situés à une distance plus petite que m de M : nombre de messages dans $S(M, m) = C_n^0 + C_n^1 + \cdots + C_n^m$. On cherche h tq l'intersection de $S(C_i, h)$ et $S(C_j, h)$ soit vide pour tous mots de code C_i et C_j différents : $h = t$. Il y a 2^k sphères $S(C, t)$ centrées sur les mots de code. Donc $2^k(C_n^0 + C_n^1 + \cdots + C_n^t) \leq 2^n)$.

Borne de Hamming

Majoration de t

1. Calculer, dans l'ordre :
   $u_0 = C_n^0 \quad u_1 = u_0 + C_n^1 \quad u_2 = u_1 + C_n^2 \quad \cdots$ Jusqu'à dépasser 2^r
2. Le premier indice m t.q. $u_m > 2^r$ est un majorant strict de t

(convergence assurée car $u_n = 2^n > 2^r$)

Borne de Hamming $d \leq 2m$

Exemple : $r = 1$

$u_0 = 1 \leq 2, u_1 = 1 + n$; or $n \geq 2$ puisque $n = k + r$. Donc $u_1 \geq 3 > 2^r$. Majoration obtenue : $1 > t$!

Code parfait

Egalité de Hamming

$C_n^0 + C_n^1 + C_n^2 + \cdots + C_n^t = 2^r \iff$ les 2^k sphères $S(C, t)$ forment une partition de $\mathbb{B}^n$.

Définition (Code parfait)

Un code est parfait s'il vérifie l'égalité de Hamming.

- $d(M, C) \leq t$
- *Expédition de C, réception de R :*
  - *soit $d(C, R) \leq t$: le message est bien corrigé*
  - *soit $d(C, R) > t$: le message est mal corrigé*
- *Un code parfait ne corrige jamais plus que t erreurs*

Exemple : code par répétition [3,1,3]

Codes linéaires

Définition (Code linéaire)

Un code $\mathcal{C}$ est linéaire lorsque $C_i \oplus C_j = C_k$ où C_i, C_j et C_k sont des mots du code.

Conséquence

Quand un code est linéaire, 0...0 est toujours un mot de code.

Exemples : test de parité, code par répétition

Facilité pour déterminer d

- La distance minimale d est le poids du mot de code non nul de plus petit poids
- $d(C_1, C_2) \geq d$

Codage linéaire

Définition (Codage linéaire)

Soit les blocs u_i. Un codage est linéaire quand l'application ϕ vérifie :

$$\phi(u_1 \oplus u_2 \oplus \cdots \oplus u_p) = \phi(u_1) \oplus \phi(u_2) \oplus \cdots \oplus \phi(u_p), \forall p \geq 1$$

Remarques

- somme de blocs = somme de leur mots de code
- par récurrence : $\phi(a \oplus b) = \phi(a) \oplus \phi(b)$

Codage linéaire (cont'd)

D'où vient l'appellation *linéaire*

- $(\mathbb{B}, \oplus, \wedge)$ est un corps
- $\mathbb{B}^n$ est un espace vectoriel de dimension n sur ce corps
- Code linéaire $=$ sous-espace vectoriel de $\mathbb{B}^n$
- Codage linéaire : application $\mathbb{B}^k \rightarrow \mathbb{B}^n$

Code ou codage linéaire : théorème

1. Codage linéaire : son code est linéaire, et 0...0 est le mot de code associé au bloc nul
2. Codage systématique, de code linéaire, est linéaire.

Simplification propre aux codes linéaires

- Codage non-linéaire : besoin des 2^k mots de code pour coder chaque bloc
- Codage linéaire : k mots de code suffisent (base canonique)

Définition (Base canonique de code)

$$\begin{array}{rcccc} b_1 & =1000 & \ldots & 00 \\ b_2 & =0100 & \ldots & 00 \\ & = & .... & ... \\ b_{k-1} & =0000 & \ldots & 10 \\ b_k & =0000 & \ldots & 01 \end{array}$$

$(b_1, b_2, \cdots, b_k)$ *forme la base canonique de* $\mathbb{B}^k$

Base canonique de code linéaire $[7, 4]$: code Cycl

$\phi(1000) = 1000101, \phi(0100) = 0100111, \phi(b_3) = 0010110$ et $\phi(b_4) = 0001011$

b_1	1000	$\phi(b_1)$	1000101
b_2	0100	$\phi(b_2)$	0100111
b_4	0001	$\phi(b_4)$	0001011
b	1101	$\phi(b)$	1101001

Représentations matricielles

Motivation

Déterminer un mot de code à partir d'un bloc = produit de matrices

Matrice génératrice (codages systématiques)

$$G = \left(\begin{array}{c|c} I_k & P \end{array} \right)$$

- I_k : matrice identité de taille k
- P (r colonnes) : matrice de parité (bits de contrôle)

Notation : $b = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \end{pmatrix} = 01001$

Matrices génératrices : exemples

Test de parité

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

Codage par répétition

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Matrices génératrices : exemples (cont'd)

Code Cycl

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

Codage linéaire défini par P !

- Codage très simple !
- Spécifier kr valeurs (bits) au lieu de $2^k r$ dans le cas g énéral des codages systématiques

Calcul facile du code d'un bloc

Le mot de code $\phi(b)$ est représenté par la matrice ligne bG (produit de b par matrice G)

Exemple : test de parité, codage de 101

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

Correction des codes linéaires

Réception de R, calcul de $\Gamma(R)$, recherche du mot e de plus petit poids dans $\Gamma(R)$, et calculer $e \oplus R$.

Objectif

Optimiser la rapidité de correction : enregistrer les actions répétées, et les lire quand on en a besoin (au lieu de les recalculer)

Théorème

1. *La relation* $\blacktriangle$ *définie sur* $\mathbb{B}^n$ *par :* $u \blacktriangle v$ *quand* $u \oplus v$ *est un mot de code, est une relation d'équivalence*
2. *Elle possède* 2^r *classes d'équivalence*
3. $\Gamma(R)$ *est la classe d'équivalence de* R
4. *Le code* $\mathcal{C}$ *est la classe de* $0\ldots0$

Correction des codes linéaires : tableau standard

Conséquence du théorème

Les ensembles $\Gamma(R)$ forment une partition de $\mathbb{B}^n$

Idée ingénieuse de stockage des $\Gamma(R)$

Construction d'une table de correction : le tableau standard

- Regrouper les mots de $\mathbb{B}^n$ par classe
- Repérer le mot de plus faible poids dans chaque classe

Structure du tableau standard

- 2^r lignes (contrôle), 2^k colonnes (blocs)
- Rangement des éléments, de telle sorte que :
  1. une ligne contient une classe d'équivalence
  2. l'élément le plus à gauche est celui de plus faible poids

Construction du tableau standard

Méthode pratique

1. Ranger les mots de $\mathbb{B}^n$ par poids croissants (si ex-aequo : ordre lexico) : obtention d'une liste $\mathcal{B}$
2. Dessiner un tableau de 2^r lignes et 2^k colonnes
3. Ranger les mots de code (taille n) dans la première ligne, en conservant l'ordre qu'ils ont dans $\mathcal{B}$ (donc 0..0 à gauche !)
4. Tout à gauche de la première ligne vide ℓ, placer le premier élément qui reste dans $\mathcal{B}$.
5. Remplir chaque case de ℓ en additionnant premier élément de la ligne, et élément au dessus de la case en première ligne.
6. Quand une ligne est remplie, recommencer la même opération ligne suivante.

Construction du tableau standard (cont'd)

Structure

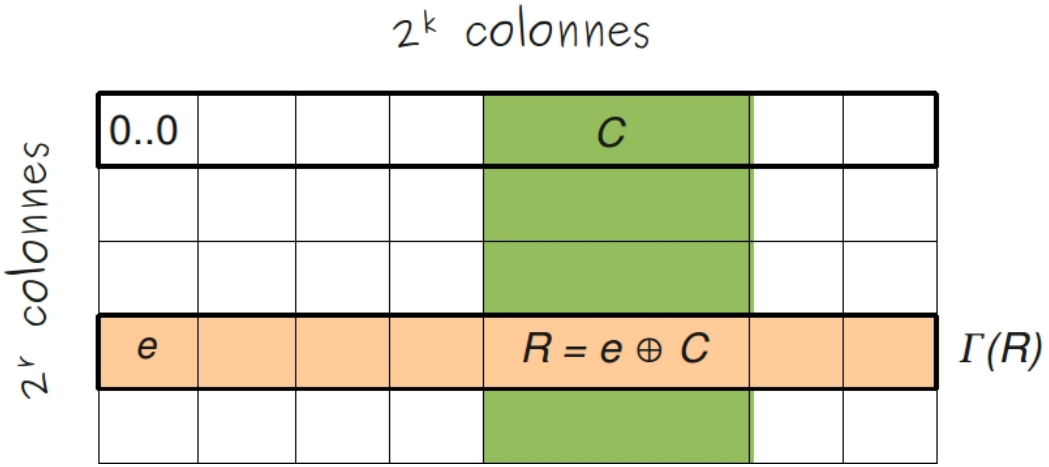


Tableau standard du code de parité $[4, 3, 2]$

0000	0011	0101	0110	1001	1010	1100	1111
0001	0010	0100	0111	1000	1011	1101	1110

Construction du tableau standard (cont'd)

Exemple du code par répétition $[3, 1, 3]$

000	111
001	110
010	101
100	011

Propriété à retenir

- Chaque ligne énonce $\Gamma(R)$ pour tous les messages reçus R de cette ligne
- Dans une ligne, les mots sont ordonnés par ordre croissant de poids !
- Si on reçoit R présent sur la ligne i colonne j, la correction sera vers $R \oplus e_i = C_j$!

Comment corriger à partir du tableau standard

Méthode pratique de correction

1. Chercher la position (i, j) du message reçu R
2. Remplacer le message R par le mot de code de la même colonne, sur la première ligne

Exemple sur code de parité $[4, 3, 2]$

(d faible)

- On remplace $R = 1000$ par $M = 1001$
- On remplace $R = 0001$ par $M = 0000$
- Pas toujours satisfaisant ! Pourquoi ne pas remplacer 0001 par 1001 ou 0101, ou...

Sur code par répétition : plus juste

Tableau standard : peut mieux faire

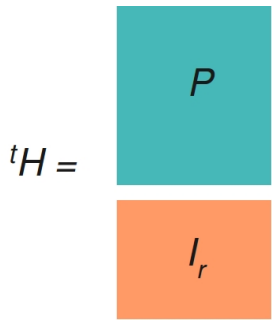
Observations

- Le tableau standrard peut être gigantesque !
- Seules les premières lignes sont exploitées (mots de faible poids)

Nécessité d'une alternative

Matrice de contrôle H

Privilégions les calculs sur le stockage



Exemples de matrices de contrôle

Parité $[4, 3, 2]$ Répétition $[3, 1, 3]$ Cycl

$$ {}^tH = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad {}^tH = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad {}^tH = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

Notion de syndrome

Définition (Syndrome du message $m = a_1 a_2 \dots a_n$)

Le syndrome de m, est $\sigma(m) = m\,{}^tH$

On a : $\sigma(m \oplus n) = \sigma(m) \oplus \sigma(n)$

Exemple : codage par répétition $[3, 1, 3]$

$$\sigma(101) = \begin{pmatrix}1 & 0 & 1\end{pmatrix} \begin{pmatrix}1 & 1\\ 1 & 0\\ 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0\end{pmatrix}$$

message	*syndrome*
011	11
101	10
110	01

Remarque

- Lignes de ${}^tH$ = tous les syndromes des messages de poids 1
- Sommes 2 à 2 des lignes de ${}^tH$ = syndromes des messages de poids 2, etc.

Utilité des syndromes

Théorème

$$M_1 \blacktriangle M_2 \iff \sigma(M_1) = \sigma(M_2)$$

Conséquence

- On attache un syndrome à chaque classe d'équivalence
- 2^r syndromes : tous les mots de r bits sont des syndromes

Correction rapide d'un message

- Remplacement du tableau standard
- Liste des syndromes : à chaque 2^r mots $\leftrightharpoons$ message de plus faible poids dont le mot est le syndrome

Etablir les liste des syndromes

Méthode pratique

1. Ranger en colonne tous les mots binaires de longueur r (ordre lexico) : les syndromes. Prévoir une seconde colonne à côté.
2. Construire $\mathcal{B}$: éléments de $\mathbb{B}^n$ rangés par poids croissants
3. Pour chaque $b \in \mathcal{B}$ (dans l'ordre) : calculer $\sigma(b)$. Si la place est libre dans la seconde colonne en face de $\sigma(b)$, y inscrire b.
4. Stop quand seconde colonne emplie.

Exemple dans le cas de Cycl

$b_1 = 1000,\ \phi(1000) = 1000101$
$b_2 = 0100,\ \phi(0100) = 0100111$
$b_3 = 0010,\ \phi(0010) = 0010110$
$b_4 = 0001,\ \phi(0001) = 0001011$

syndrome	*message*
000	0000000
001	0000001
010	0000010
011	0001000
100	0000100
101	1000000
110	0010000
111	0100000

Déroulons l'exemple

Mots 2^r	
000	
001	
010	
011	
100	
101	
110	
111	

Calculons $\sigma(0000000)$

Messages 2^n

0000000
0000001
0000010
0000100
0001000
0010000
0100000
1000000
0000011
...
1111111

Calcul de $\sigma(0000000)$

$$\sigma(0000000) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \end{pmatrix}^t = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

Déroulons l'exemple (cont')

Mots 2^r	
000	0000000
001	
010	
011	
100	
101	
110	
111	

Calculons $\sigma(0000001)$

Messages 2^n

0000000
0000001
0000010
0000100
0001000
0010000
0100000
1000000
0000011
...
1111111

Calcul de $\sigma(0000001)$

$$\sigma(0000001) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \end{pmatrix}^t = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

Déroulons l'exemple (cont')

Mots 2^r

000	0000000
001	0000001
010	
011	
100	
101	
110	
111	

Calculons $\sigma(0000010)$...

Messages 2^n

0000000
0000001
0000010
0000100
0001000
0010000
0100000
1000000
0000011
...
1111111

Calcul de $\sigma(0000010)$

$$\sigma(0000010) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \end{pmatrix}^t = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$

Déroulons l'exemple (cont')

Mots 2^r	
000	0000000
001	0000001
010	0000010
011	
100	
101	
110	
111	

Calculons $\sigma(0000100)$...

Messages 2^n

0000000
0000001
0000010
0000100
0001000
0010000
0100000
1000000
0000011
...
1111111

Calcul de $\sigma(0000010)$

$$\sigma(0000100) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \end{pmatrix}^t = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

Déroulons l'exemple (cont')

Mots 2^r	
000	0000000
001	0000001
010	0000010
011	
100	0000100
101	
110	
111	

Calculons $\sigma(0001000)$...

Messages 2^n

0000000
0000001
0000010
0000100
0001000
0010000
0100000
1000000
0000011
...
1111111

Corriger avec les syndromes

Principe

Avec R reçu : on le corrige en lui ajoutant l'élément de $\Gamma(R)$ de plus petit poids, donc l'élément de $\mathbb{B}^n$ qui a le même syndrome que R avec le plus petit poids possible.

Méthode pratique pour corriger R

1. Calculer $\sigma(R)$
2. Repérer S : le message associé à $\sigma(R)$ dans la liste.
3. Corriger R en le remplaçant par $R \oplus S$

Exemple avec Cycl

Soit $R = 1010101$,

$$\sigma(R) = R^t H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$$

Avant dernière ligne de la liste des syndromes : $M = 0010000$

$$C = 1010101 \oplus 0010000 = 1000101$$

Code parfait

Un code correcteur parfait doit corriger parfaitement les messages qui n'ont qu'une erreur : $t \geq 1$ ou $d \geq 3$

Théorème

Un codage linéaire corrige de façon certaine toutes les erreurs de poids 1 $\Longleftrightarrow$ *sa matrice* ${}^tH$ *a toutes ses lignes distinctes et non nulles.*

Les lignes de ${}^tH$ sont les syndromes des messages de poids 1 : pas de mot de code de poids 1 $\Longleftrightarrow$ pas de ligne nulle dans ${}^tH$

Définition (Code de Hamming)

Un code de Hamming est un code linéaire $[2^r - 1, 2^r - 1 - r]$ *dont la matrice* ${}^tH$ *est constituée de* tous *les mots binaires non nuls de longueur* r.

- $d = 3$
- *Code parfait*