Codage des mots de poids constant

Nicolas Sendrier

INRIA Rocquencourt, projet CODES

Journées Codage et Cryptographie, Carcans, 17-21 mars 2008

I. Introduction

Contexte : la cryptographie basée sur les codes

Soit $\mathcal{C}$ un code binaire $[n, k]$

- G une matrice géneratrice de $\mathcal{C}$, de taille $k \times n$
- H une matrice de parité de $\mathcal{C}$, de taille $(n-k) \times n$

Nous noterons $W_{n,t}$ l'ensemble des mots de longueur n et de poids de Hamming t (typiquement $t \ll n$).

En cryptographie basée sur les codes on utilise deux primitives :

$$\begin{array}{rccc|rccc} \Psi_G : & \{0,1\}^k \times W_{n,t} & \longrightarrow & \{0,1\}^n & S_H : & W_{n,t} & \longrightarrow & \{0,1\}^{n-k} \\ & (x, e) & \longmapsto & xG + e & & (x, e) & \longmapsto & eH^\top \end{array}$$

Toutes deux sont « à sens unique »

Principaux systèmes

Systèmes a clé publique

- McEliece [1978] : le chiffrement utilise $\Psi_G : \{0,1\}^k \times W_{n,t} \to \{0,1\}^n$, mais le mot de $W_{n,t}$ est choisi au hasard.

  Néanmoins les conversions sémantiquement sûres nécessitent une fonction inversible $\{0,1\}^\ell \to W_{n,t}$
- Niederreiter [1986] : le chiffrement utilise $S_H : W_{n,t} \to \{0,1\}^{n-k}$
- Signature [Asiacrypt 2001, Courtois, Finiasz, S.] : la signature est un mot de $W_{n,t}$

Principaux systèmes (2)

Il existe également des systèmes symétriques utilisant les codes.

- générateur pseudo-aléatoire [Eurocrypt 96, Fischer, Stern]
- fonction de hachage [MyCrypt 2005, Augot, Finiasz, S.]
- chiffrement par flot [ISIT 2007, Gaborit, Lauradoux, S.]

Tous utilisent la fonction de syndrome $S_H : W_{n,t} \to \{0,1\}^{n-k}$ et les deux derniers des mots réguliers.

Les mots réguliers sont obtenus en concaténants t mots de poids 1 et de longueur n/t. On a une fonction $\{0,1\}^{\ell} \to W_{n,t}$ avec $\ell = t{\cdot}\log_2(n/t)$. Le coût du codage est négligeable.

Problème : impact possible sur la sécurité.

Vitesse – Ordres de grandeur

Coût des primitives cryptographiques

	cycles/octet sur Pentium 4	
	chiffrement	déchiffrement
Chiffrement par flot (SOSEMANUK)	5-10	5-10
Chiffrement par bloc (AES)	15-30	15-30
RSA 2048 (SSL)	2300	235K
RSA 1024 (SSL)	1900	95K
NTRU 787 (source EBATS)	10500	19K
McEliece* $(2048, 32)$	510	5.5K
Niederreiter* $(2048, 32)$	63	45K

* sans codage en mots de poids constant

Plan

I. Introduction
II. Formalisation du problème
III. Quelques éléments de codage de source
IV. Première solution : méthode énumérative
V. Deuxième solution : méthode récursive
VI. Troisième solution : méthode dichotomique
VII. Conclusion

II. Formalisation du problème

Codage de l'information par des mots de poids constant

Le but est de coder de l'information (binaire) en mots de $W_{n,t}$. Donc on cherche une application injective

$$\phi_{n,t} : \{0,1\}^{\ell} \longrightarrow W_{n,t}$$

facile à calculer et à inverser.

C'est souvent le facteur limitant pour le temps de calcul en cryptographie (chiffrement Niederreiter, flot, hachage).

On veut ℓ le plus grand possible, idéalement $\ell = \left\lfloor \log_2 \binom{n}{t} \right\rfloor$.

Sinon, perte de sécurité possible : jusqu'à $\dfrac{\binom{n}{t}}{2^{\ell}}$

Simuler une source à l'aide d'une séquence binaire

Soit une source finie (un alphabet fini X et une loi de probabilité sur X). Soit un code $f : X \to \{0,1\}^*$ de cette source. Le codage associé à f est défini par

$$\begin{array}{rccc} F : & X^* & \longrightarrow & \{0,1\}^* \\ & (x_1, \ldots, x_L) & \longmapsto & f(x_1) \parallel \cdots \parallel f(x_L) \end{array}$$

- Le code est *à décodage unique* si F est injectif.
- Le code est *irréductible* si F est bijectif.
- Un code optimal est irréductible.

Pour simuler la source on utilise F^{-1} avec en entrée des bits équidistribués. Donc il faudra un F bijectif (surjectif suffit). Si le code est optimal, c'est mieux...

Simuler une source à l'aide d'une séquence binaire

Si le code f est prefixe et que son codage associé F est bijectif (l'optimalité de f pour n'importe quelle loi de probabilité suffit), alors on pourra définir

$$\phi : \{0,1\}^* \longrightarrow X$$

dont l'entrée est binaire et de longueur variable.

Pour obtenir une longueur fixe, il faut $\ell \leq \min_{x \in X} |f(x)|$

$$\begin{array}{rccl} \phi : & \{0,1\}^\ell & \longrightarrow & X \\ & s & \longmapsto & x \text{ tel que } f(x) = s \parallel 0^i, i \geq 0 \end{array}$$

On engendre (uniformément) un sous-ensemble de X de taille 2^ℓ.

Codage des mots de poids constant

Soit la source $W_{n,t}$ munie d'une loi uniforme ($2^\ell \leq \binom{n}{t} < 2^{\ell+1}$).

Un code optimal $f : W_{n,t} \rightarrow \{0,1\}^*$ de cette source comporte

- $2^{\ell+1} - \binom{n}{t}$ mots de longueur ℓ
- $2\binom{n}{t} - 2^{\ell+1}$ mots de longueur $\ell + 1$

En utilisant le décodeur correspondant, on peut associer à tout $s \in \{0,1\}^\ell$ un élément de $W_{n,t}$:

- si s est un mot de code $\rightarrow f^{-1}(s)$
- sinon $\rightarrow f^{-1}(s \parallel 0)$

Codage des mots de poids constant - Exemple

$$n = 5,\ t = 2,\ \ell = 3,\ 2^3 < \binom{5}{2} = 10 < 2^4$$

f			ϕ		
11000	$\longrightarrow$	000	000	$\longrightarrow$	11000
10100	$\longrightarrow$	001	001	$\longrightarrow$	10100
10010	$\longrightarrow$	010	010	$\longrightarrow$	10010
10001	$\longrightarrow$	011	011	$\longrightarrow$	10001
01100	$\longrightarrow$	100	100	$\longrightarrow$	01100
01010	$\longrightarrow$	101	101	$\longrightarrow$	01010
01001	$\longrightarrow$	1100	110	$\longrightarrow$	01001
00110	$\longrightarrow$	1101			
00101	$\longrightarrow$	1110	111	$\longrightarrow$	00101
00011	$\longrightarrow$	1111			

III. Quelques éléments de codage de source

Source finie uniformément distribuée

Source : alphabet fini $X = \{0, 1, \ldots M-1\}$ muni d'une loi uniforme.

Soit $2^\ell \leq M < 2^{\ell+1}$. Un code binaire optimal aura

- $d = 2^{\ell+1} - M$ mots de longueur ℓ
- $M - d = 2M - 2^{\ell+1}$ mots de longueur $\ell + 1$

Par exemple $\varphi_M(i) = \begin{cases} [i]^2_\ell & \text{si } 0 \leq i < d \\ [i+d]^2_{\ell+1} & \text{si } d \leq i < M \end{cases}$

où $[i]^2_\ell$ est l'écriture en base 2 de i sur ℓ bits avec les bits les plus significatifs en tête.

Décodage de $s = (s_1, \ldots, s_\ell, s_{\ell+1}, \ldots)$. Soit $s = \sum_{1 \leq i \leq \ell} s_i 2^{\ell - i}$

- si $s < d$ alors on retourne s
- sinon on retourne $2s + s_{\ell+1} - d$

Codage arithmétique - Principe

On considère une source produisant des lettres d'un alphabet X. La loi de la source est éventuellement complexe (processus stochastique).

Le mot de code correspondant à un texte $(x_1, \ldots, x_L) \in X^L$ est égal aux ℓ bits les plus significatifs du développement en base deux$^{(*)}$ de

$$S(x_1, \ldots, x_L) = \sum_{(y_1, \ldots, y_L) < (x_1, \ldots, x_L)} P(y_1, \ldots, y_L)$$

avec

$$\ell = \left\lceil \log_2 \frac{1}{P(x_1, \ldots, x_L)} \right\rceil + 1$$

C'est un code préfixe non optimal de X^L.

$^{(*)}$ si $p \in [0, 1[$ alors $p = \sum_{i>0} p_i 2^{-i}$ (ou encore $p = 0.p_1 p_2 p_3 \ldots$)

Codage arithmétique - Algorithme

Pour tout $n \leq L$ le texte $(x_1, \ldots, x_n)$ est représenté par l'intervalle $[a_n, a_n + \delta_n[$ avec $a_n = S(x_1, \ldots, x_n)$ et $\delta_n = P(x_1, \ldots, x_n)$

Ces intervalles sont définis par récurrence : $a_0 = 0, \delta_0 = 1$ et

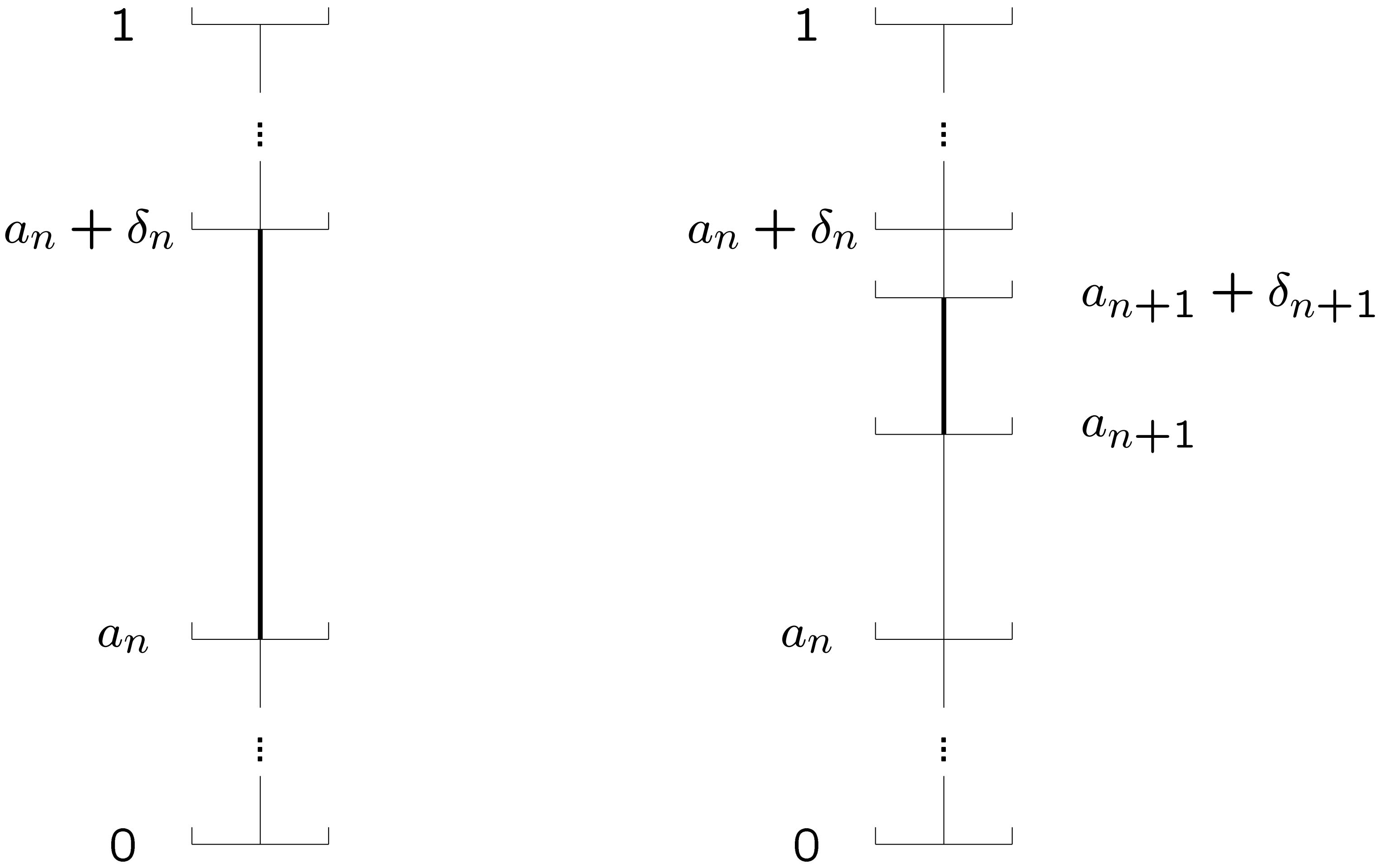
$$\left\{ \begin{array}{lcl} a_0 & = & 0 \\ \delta_0 & = & 1 \end{array} \right. \text{ et } \left\{ \begin{array}{lcl} a_{n+1} & = & a_n + S_{n+1}(x_{n+1}) \cdot \delta_n \\ \delta_{n+1} & = & P_{n+1}(x_{n+1}) \cdot \delta_n \end{array} \right.$$

où $P_{n+1}(x) = P(x \mid x_1, \ldots, x_n)$ et $S_{n+1} = \sum_{y<x} P_{n+1}(y)$.

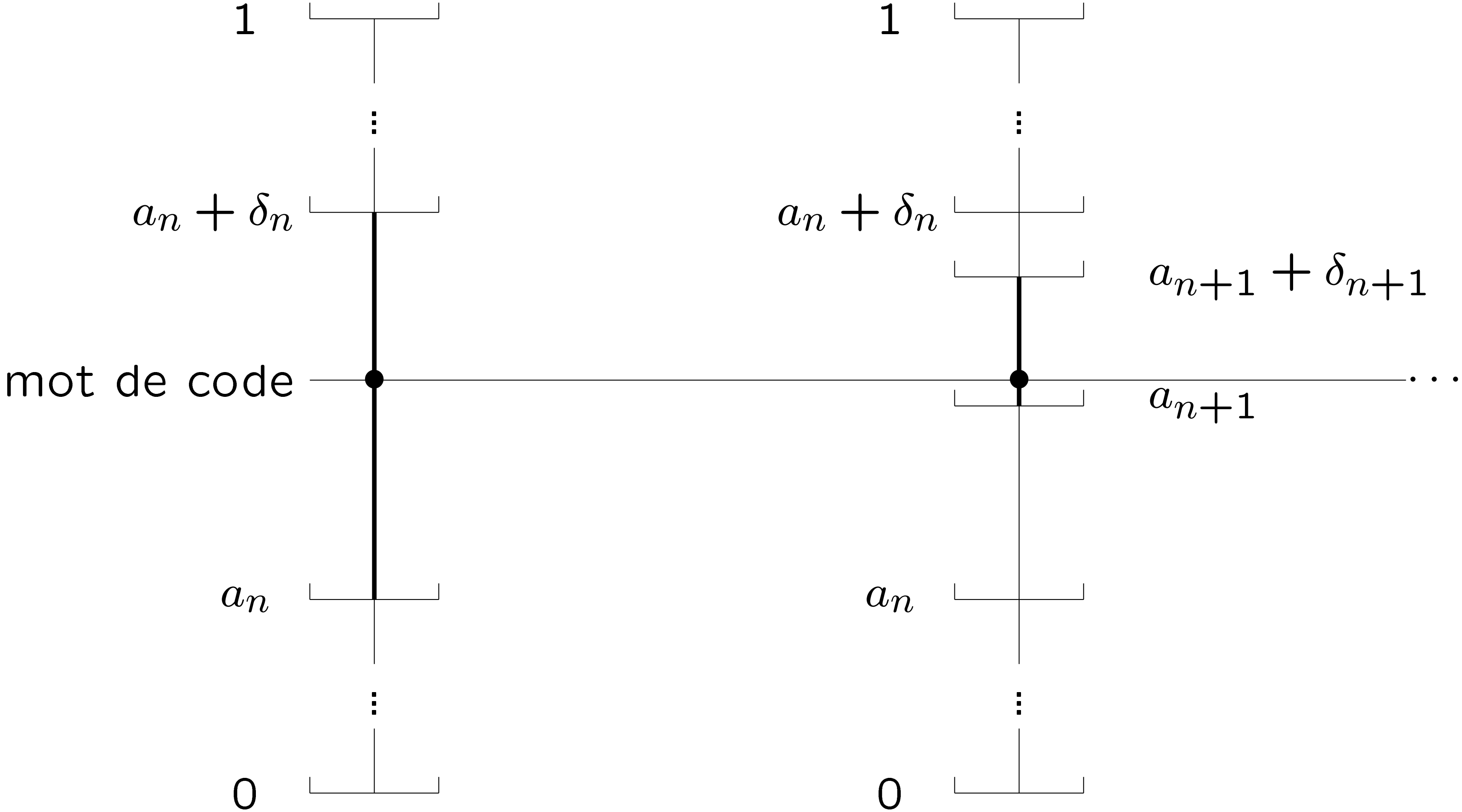
Le codage du texte $(x_1, \ldots, x_L)$ sera égal aux $\lceil -\log_2 \delta_L \rceil + 1$ premiers bits du développement en base deux de $a_L + \delta_L/2$.

Pour tout L, on obtient un code de X^L prefixe mais pas irréductible.

Codage arithmétique - Algorithme, codage



Codage arithmétique - Algorithme, décodage



Codage arithmétique - Implémentation

Codeur $\mathcal{C}$	**Décodeur** $\mathcal{D}$
entrée : $x_1, \ldots, x_L$	entrée : $s \in \{0,1\}^\ell$
sortie : $s \in \{0,1\}^\ell$	sortie : $x_1, \ldots, x_L$
temps n : lit (x_n, P_n)	temps n : lit P_n, écrit x_n
temps $L+1$: écrit $s \in \{0,1\}^\ell$	

La séquence binaire s, et sa longueur, sont une fonction de l'état. Bien sûr, on a

$$\mathcal{C}(x_1, \ldots, x_L) = s \Rightarrow \mathcal{D}(s) = (x_1, \ldots, x_L)$$

Witten, Neal et Cleary ont décrit une implémentation en précision fixe. Elle est légèrement sous-optimale mais ça ne change rien en pratique

IV. Première solution : méthode énumérative

Méthode énumérative – Codage

Un mot de $W_{n,t}$ est représenté par les indices $0 \leq i_1 < i_2 < \ldots < i_t < n$ désignant les coordonnées non nulles. Soit

$$\begin{array}{rccc} \theta : & W_{n,t} & \longrightarrow & \left[0, \binom{n}{t}\right[\\ & (i_1, \ldots, i_t) & \longmapsto & \binom{i_1}{1} + \binom{i_2}{2} + \cdots + \binom{i_t}{t} \end{array}$$

On montre facilement que θ est strictement croissant (ordre lexicographique pour $W_{n,t}$). L'image du plus petit élément vaut 0 et celle du plus grand $\binom{n}{t} - 1$.

Pour obtenir des mots binaires, on utilise le codeur φ_M pour $M = \binom{n}{t}$

$$f(e) = \varphi_M(\theta(e))$$

Méthode énumérative – Décodage

Il existe une formule d'inversion pour les coefficients binomiaux.

$$\left(x = \binom{i}{t}\right) \Leftrightarrow \left(i = X + \frac{t-1}{2} + \frac{t^2-1}{24}\frac{1}{X} + \mathcal{O}\left(\frac{1}{X^3}\right)\right) \text{ où } X = (t!x)^{1/t}$$

À partir de cette formule, on peut construire une fonction $\mathtt{inv\_bino}(x,t)$ dont la valeur est l'unique entier i tel que $\binom{i}{t} \leq x < \binom{i+1}{t}$

entree : $x \in \left[0, \binom{n}{t}\right[$
sortie : t entiers $0 \leq i_1 < i_2 < \ldots < i_t < n$
$j \leftarrow t$
tantque $j > 0$
  $i_j \leftarrow \mathtt{inv\_bino}(x,j)$; $x \leftarrow x - \binom{i_j}{j}$; $j \leftarrow j-1$

En pratique le coût de `inv_bino` n'est pas pénalisant.

Méthode énumérative – Temps de calcul

$$\ell = \left\lfloor \log_2 \binom{n}{t} \right\rfloor \qquad \phi_{n,t} : \{0,1\}^\ell \to W_{n,t}$$

					avec précalcul	
n	t	ℓ	$\phi_{n,t}$	$\phi_{n,t}^{-1}$	$\phi_{n,t}$	$\phi_{n,t}^{-1}$
2048	32	233	4200	3000	1500	570
2048	40	280	4600	3600	1650	660
4096	22	193	3100	2100	1300	530
4096	45	353	5800	4900	1650	760

Cycles/octet sur Pentium 4

Comment faire mieux ?

Changer de modèle de source. Par exemple

$$\begin{array}{cccccc} i_1 & i_2 & & & & i_t \\ \boxed{0\cdots010\cdots0} & \boxed{0\cdots010\cdots0} & \cdots & & \cdots & \boxed{0\cdots010\cdots0} \\ \longleftrightarrow & \longleftrightarrow & & & & \longleftrightarrow \\ n/t & n/t & & & & n/t \end{array}$$

Chaque position i_j est choisie uniformément dans l'intervalle

$$\left[(j-1)\frac{n}{t}, j\frac{n}{t}\right[$$

$\longrightarrow$ mots réguliers. Pas toujours satisfaisant.

$\ell = t\log_2(n/t)$ alors que $\log_2\binom{n}{t} \approx t\log_2(n/t) + t/\ln(2)$

V. Deuxième solution : méthode récursive

Méthode récursive – Principe

On s'intéresse à la longueur des séquences de '0'

$$e = \underbrace{\boxed{0\cdots0}}_{\delta}\ 1\ \underbrace{\boxed{0\cdots0}\ 1\ \cdots\ \cdots\ 1\ \boxed{0\cdots0}\ 1\ \boxed{0\cdots0}}_{e'}$$

Au mot e choisi uniformément dans $W_{n,t}$ correspond un couple (δ, e') avec $\delta \in [0, n-t[$ obéissant à la loi

$$P_{n,t}(\delta) = \frac{\binom{n-\delta-1}{t-1}}{\binom{n}{t}}$$

et $e' \in W_{n-\delta-1,t-1}$ obéissant à une loi uniforme.

$$\Psi_{n,t}(e) = f_{n,t}(\delta) \parallel \Psi_{n-\delta-1,t-1}(e')$$

Méthode récursive – Modèle simplifié

$$e = \underbrace{\boxed{0 \cdots 0}}_{\delta}\ 1\ \underbrace{\boxed{0 \cdots 0}\ 1\ \cdots\ \cdots\ 1\ \boxed{0 \cdots 0}\ 1\ \boxed{0 \cdots 0}}_{e'}$$

L'entier δ reste compliqué à coder, mais on va faire une approximation.
On choisit d (fonction de n et t) tel que

$$P_{n,t}(\delta \geq d) = \sum_{\delta \geq d} \frac{\binom{n-\delta-1}{t-1}}{\binom{n}{t}} = \frac{\binom{n-d}{t}}{\binom{n}{t}} \approx \frac{1}{2}$$

et on considère une loi $P'_{n,t}$ telle que

$$P'_{n,t}(\delta \geq d) = \frac{1}{2} \text{ et } P'_{n,t}(\delta) = \frac{1}{d} \text{ pour } \delta \in [0, d[$$

Méthode récursive – Algorithme

$$e = \boxed{0 \cdots 0}\ 1\ \underbrace{\boxed{0 \cdots 0}\ 1\ \cdots\ \cdots\ 1\ \boxed{0 \cdots 0}\ 1\ \boxed{0 \cdots 0}}_{e'}$$

$$\underbrace{\boxed{0 \cdots 0}}_{\delta}$$

$$\Psi_{n,t}(e) = \begin{cases} 0 \parallel \Psi_{n-d,t}(\delta - d, e') & \text{si } \delta \geq d \\ 1 \parallel \varphi_d(\delta) \parallel \Psi_{n-\delta-1,t-1}(e') & \text{sinon} \end{cases}$$

et φ_d est un codage optimal de $[0, d[$ muni d'une loi uniforme.

L'entier d doit vérifier $2^{\binom{n-d}{t}} \approx \binom{n}{t}$. Il existe une assez bonne approximation par une formule close

$$d \approx \left(1 - \frac{1}{2^{1/t}}\right)\left(n - \frac{t-1}{2}\right)$$

Méthode récursive – Implémentation

- Pas besoin de calcul en précision arbitraire.
- Le calcul de d a un coût négligeable en pratique car on peut mettre en table les $(1 - 1/2^{1/t})$.
- Si on se « trompe » de valeur pour d, l'impact est faible

  $\longrightarrow$ Il existe une optimisation consistant à choisir pour d une puissance de 2, cela simplifie le codage par φ_d et accélère siginicativement le calcul.

La longueur moyenne du codage est proche de $\log_2 \binom{n}{t}$, mais les mots de code ont une longueur variable.

Méthode récursive – Chiffres

		cycles/octet		longueurs			
n	t	$\phi_{n,t}$	$\phi_{n,t}^{-1}$	min.	max.	moy.	$\log_2 \binom{n}{t}$
2048	32	530	500	226	241	233.0	233.9
		277	**275**	**219**	**244**	**231.6**	
2048	40	540	510	271	288	279.1	280.2
		280	**278**	**265**	**289**	**278.4**	
4096	22	470	450	187	200	193.3	193.9
		250	**237**	**182**	**200**	**192.9**	
4096	45	500	470	343	361	352.0	353.3
		268	**259**	**335**	**362**	**251.0**	

Sur Pentium 4, en gras la version rapide (d puissance de 2)

VI. Troisième solution : méthode dichotomique

Méthode dichotomique – Principe

Chaque mot de $W_{n,t}$ est découpé en deux parties de même taille

$$e = \overbrace{\boxed{\text{poids } i}}^{e^{(G)}}_{\longleftrightarrow\; n/2}\ \overbrace{\boxed{\text{poids } t-i}}^{e^{(D)}}_{\longleftrightarrow\; n/2}$$

La loi de i est la suivante : $P_{n,t}(i) = \dfrac{\binom{n/2}{i}\binom{n/2}{t-i}}{\binom{n}{t}}$

Pour coder e choisi uniformément dans $W_{n,t}$

– coder $i \in [0, t]$ selon la loi $P_{n,t}$

– coder $e^{(G)} \in W_{n/2,i}$ selon une loi uniforme

– coder $e^{(D)} \in W_{n/2,t-i}$ selon une loi uniforme

Méthode dichotomique – Algorithme

fonction `coder`
  entree : $e \in W_{n,t}$
  sortie : $s \in \{0,1\}^*$
    $\mathcal{C}$.`init`()
    `coder_rec`$(n, t, e, \mathcal{C})$
    $s \leftarrow \mathcal{C}$.`fermer`()

procedure `coder_rec`
  entree : $n, t, e \in W_{n,t}, \mathcal{C}$
    $i \leftarrow$ `poids`$(e^{(G)})$
    $\mathcal{C}$.`lire`$(i, P_{n,t})$
    `coder_rec`$(n/2, i, e^{(G)}, \mathcal{C})$
    `coder_rec`$(n/2, t-i, e^{(D)}, \mathcal{C})$

- $\mathcal{C}$.`init`() initialise un codeur arithmétique $\mathcal{C}$.
- $\mathcal{C}$.`lire`$(i, P_{n,t})$ lit un entier i et une loi $P_{n,t}$ et modifie l'état du codeur.
- $\mathcal{C}$.`fermer`() retourne le mot de code correspondant à l'état et ferme le codeur arithmétique.

Méthode dichotomique – Implementation

- $n = 2^m$
- Mise en table des $P_{n,t}$ mais avec $i \in [u, t-u]$

  Taille raisonnable car $n = 2^m$
- sous-optimal pour 2 raisons
  - calculs en précision finie (WNC)
  - approximation des $P_{n,t}$
- On peut calculer l'erreur maximale (due à la sous-optimalité) par programmation dynamique

  $\rightarrow$ borne sur la perte maximale d'information (en pratique 0.2 à 0.3 bits)
- Si $\binom{n}{t} < 2^{32}$, on utilise la méthode énumérative (avec précalculs).
- Coder $W_{n',t}$ avec $n' = \dfrac{n}{2^u}$. On a $t + \log_2 \binom{n/2}{t} \approx \log_2 \binom{n}{t}$.

Méthode dichotomique – Temps de calcul

n	t	ℓ	$\log_2 \binom{n}{t}$	$\phi_{n,t}$	$\phi_{n,t}^{-1}$
2048	32	232	233.9	520	400
2048	40	280	280.2	600	440
4096	22	192	193.9	410	320
4096	45	352	353.3	510	380

Cycles/octet sur Pentium 4

VII. Conclusion

Conclusion

Le codage des mots de poids constant a un coût significatif dans tous le cas, parfois dominant.

Différents compromis sont possibles :

- Méthode énumérative : lent, longueur fixe
- Méthode récursive : rapide, longueur variable
- Méthode dichotomique : rapide, longueur fixe, $n = 2^m$, tables (10 à 15 Ko)

Autres possibilités :

- Méthode récursive sans approximations avec codage arithmétique
- Méthode dichotomique avec des approximations plus grossières (tables plus petites), longueur quelconque