UQAR

Codes économiques réexaminés : généralisations, bornes et algorithmes

Steven Pigeon
Professeur
Département de mathématiques, informatique et génie
steven_pigeon@uqar.ca

Université du Québec à Rimouski

29 mai 2026

Énoncé du problème

Présentés par Eastman *et al.* [1, 2], les codes économiques permettent d'encoder des valeurs $0 \leqslant v < n$, iid uniformément lorsque n n'est pas une puissance de deux, grâce à un code particulièrement simple, un cas particulier des codes de Huffman [3].

Le paradigme pour les codes économiques est une faible puissance de calcul/débit extrêmement élevé, possiblement au détriment de la compression car on fait une hypothèse simplificatrice d'uniformité.

Une remarque

Des résultats qui auraient dû être obtenus il y a 40 ans ?

Les codes sont mentionnés à plusieurs reprises et sont utilisés dans divers logiciels et protocoles de compression de données, mais...

- Les résultats se limitent toujours à un alphabet de codage binaire,
- Bien qu'on ne fasse pas intervenir de techniques particulièrement ardues, les dérivations n'apparaissent pas dans la littérature.
- *Crede mihi, frater* n'est pas une bonne façon de faire de la science.

Objectifs

Que veut-on montrer ?

- Comment construire le code
- La longueur moyenne des codes
- ★ L'inefficacité maximale
- ★ Le gain spécifique et le gain espéré
- Les procédures de codage et de décodage

Hypothèses de départ

Nous supposons

- Une base $b \geqslant 2$, *
- Le nombre de valeurs $n = b^k + r$ quelconque, mais tel que $b - 1 \,|\, n - 1$, **
- Les valeurs $0 \leqslant v < n$ à coder sont iid uniformément.

* Dans ce qui suit, nous dirons « base » pour la taille de l'alphabet de codage et « chiffre » pour un symbole.

** Nécessaire pour la procédure de Huffman.

Un cas particulier

Puisque la distribution est supposée uniforme, il nous faut construire un arbre de codes complet où toutes les feuilles sont sur au plus deux profondeurs adjacentes.

C'est un cas particulier des codes de Huffman [3].

Nous allons construire l'arbre de codes grâce à la procédure de Huffman. *

* Nous verrons que les procédures d'encodage et de décodage nous dispensent d'utiliser explicitement un arbre !

Créer l'arbre de codes

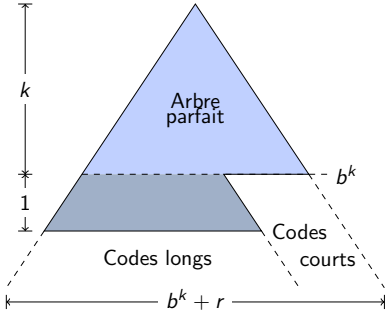
Pour construire l'arbre de codes, nous utiliserons la décomposition

$$n = b^k + r,$$

qui est unique si $0 \leqslant r < b^{k+1} - b^k = b^k(b-1)$.

Cette décomposition nous permettra de trouver le nombre nécessaire de fusions pour réduire l'arbre de codes à un arbre parfait.

Créer l'arbre de codes



Les codes « longs » ont un chiffre de plus que les codes « courts ».

Créer l'arbre de codes

Avec la procédure de Huffman,

- Nous commençons avec n nœuds, tous de poids égal.
- Nous retirons les b nœuds de poids plus faibles, les fusionnons, et ajoutons un nouveau nœud dont le poids est la somme des nœuds fusionnés.
- À chaque itération, le nombre de nœuds actifs est réduit de $b - 1$.

Nous répétons jusqu'à ce que nous ayons b^k nœuds actifs.

Créer un arbre parfait

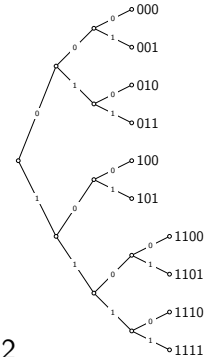
Le nombre de fusions nécessaires pour atteindre un arbre parfait est $r = m(b-1)$, soit encore $m = \frac{r}{b-1}$.

Puisque ces m fusions représentent chacune b feuilles, nous aurons

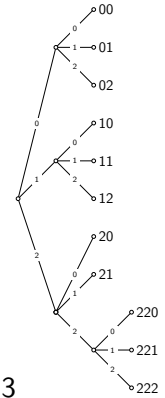
- $l = bm = \dfrac{br}{b-1}$ codes longs,
- et $s = n - bm = b^k - \dfrac{r}{b-1}$ codes courts.

(Cette valeur s nous servira de seuil pour les procédures rapides d'encodage et de décodage.)

Deux exemples



$b = 2$



$b = 3$

Longueur espérée

Maintenant que nous connaissons le nombre de codes courts et de codes longs, la longueur espérée s'obtient sans peine :

$$\begin{aligned}\bar{L}_b(n) = \bar{L}_b(b^k + r) &= \frac{sk + l(k+1)}{b^k + r} \\ &= k + \frac{br}{(b-1)(b^k + r)}.\end{aligned}$$

Un résultat donné sans justification pour $b = 2$ dans [5], mais démontré dans [4].

Longueur espérée

Maintenant que nous connaissons le nombre de codes courts et de codes longs, la longueur espérée s'obtient sans peine :

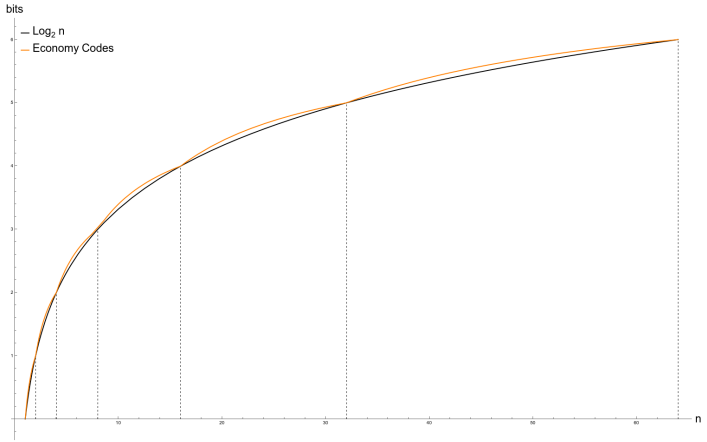
$$\begin{aligned}\bar{L}_2(n) = \bar{L}_2(2^k + r) &= \frac{sk + l(k+1)}{2^k + r} \\ &= k + \frac{2r}{2^k + r}.\end{aligned}$$

Un résultat donné sans justification pour $b = 2$ dans [5], mais démontré dans [4].

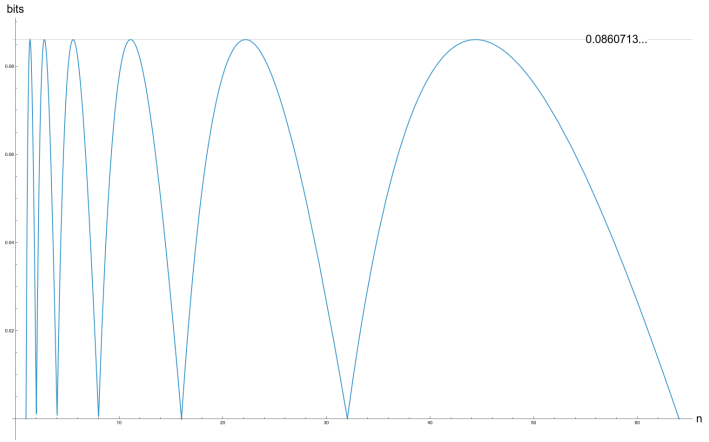
Inefficacité

Puisqu'un code économique est un arbre b-aire de codes utilisant un nombre entier de chiffres pour chaque code, il contiendra nécessairement une part d'inefficacité. Un code idéal utiliserait *exactement* $\log_b n$ chiffres pour encoder une valeur.

Inefficacité : $\bar{L}_2(n)$ vs $\log_2 n$



Inefficacité : $\bar{L}_2(n) - \log_2 n$



Inefficacité maximale

L'inefficacité est donc

$$0 < \bar{L}_b(n) - \log_b n < 1$$

et puisque c'est convexe en r, on peut trouver l'inefficacité maximale en résolvant

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(\bar{L}_b(n) - \log_b(n)\right) = 0$$

pour r. On trouve

$$r^* = \frac{b^k(b(\ln b - 1) + 1)}{b - 1}.$$

inefficacité maximale

Nous trouvons que le « pire » $b^k < n < b^{k+1}$ est

$$n^* = b^k + r^* = \frac{b^{k+1} \ln b}{b-1}.$$

L'inefficacité maximale est enfin

$$\bar{L}_b(n^*) - \log_b n^* = \frac{1}{b-1} + \frac{\ln(b-1) - \ln\ln b - 1}{\ln b}.$$

Pour le cas particulier $b = 2$, on trouve que $\bar{L}_2(n^*) - \log_2 n^* = 0.086071332\ldots$

Gain spécifique

Si nous utilisons un code naïf de $\lceil \log_b n \rceil$ chiffres pour encoder nos valeurs, le gain (le nombre de chiffres épargnés) d'un code économique est

$$\begin{aligned} \lceil \log_b n \rceil - \bar{L}_b(n) &= (k+1) - \left(k + \frac{br}{(b-1)(b^k + r)} \right) \\ &= 1 - \frac{br}{(b-1)(b^k + r)}. \end{aligned}$$

Gain espéré

Alors, pour un n quelconque*, on peut se demander quel sera le gain espéré.

Nous avons :

$$\mathbb{E}\left[\lceil \log_b n \rceil - \bar{L}_b(n)\right] = \frac{1}{b^k - 1} \sum_{\substack{r=b-1,2(b-1),\\ \ldots,(b^k-1)(b-1)}} 1 - \frac{br}{(b-1)(b^k + r)}$$

$$= 1 - \frac{b}{(b^k - 1)(b-1)} \sum_{j=1}^{b^k-1} \frac{j(b-1)}{b^k + j(b-1)}.$$

* Mais toujours tel que $b - 1 \,|\, n - 1$.

Gain espéré

Remarquons que

$$\sum_{r=1}^{u} \frac{r}{v+r} = u - v \sum_{r=1}^{u} \frac{1}{v+r},$$

puis que

$$\sum_{j=a}^{b} \frac{1}{1+js} = \frac{1}{s}\left(H(b+\frac{1}{s}) - H(a-1+\frac{1}{s}) \right),$$

où $H(m) = \psi(m) + \gamma \approx \ln m + \gamma$ est le $m^{\text{ième}}$ nombre harmonique généralisé, $\psi(m)$ la fonction digamma et $\gamma = 0.577215644\ldots$ la constante Euler-Mascheroni.

Gain espéré

Poursuivons :

$$\begin{aligned}\mathbb{E}\left[\lceil \log_b n \rceil - \bar{L}_b(n)\right] &= 1 - \frac{b}{(b^k-1)(b-1)} \sum_{j=1}^{b^k-1} \frac{j(b-1)}{b^k + j(b-1)} \\ &= 1 - \frac{b}{(b^k-1)(b-1)} \left((b^k-1) - \frac{b^k}{b-1} \left(H\left(\frac{b^{k+1}}{b-1} - 1\right) - H\left(\frac{b^k}{b-1}\right)\right)\right) \\ &= \frac{b^{k+1}}{(b^k-1)(b-1)^2} \left(H\left(\frac{b^{k+1}}{b-1} - 1\right) - H\left(\frac{b^k}{b-1}\right)\right) - \frac{1}{b-1}.\end{aligned}$$

Gain espéré

Enfin, nous avons que

$$\lim_{k\to\infty} \mathbb{E}\left[\lceil \log_b n \rceil - \bar{L}_b(n)\right] = \frac{b\ln b - (b-1)}{(b-1)^2}.$$

Pour $b = 2$, nous découvrons que le gain espéré est $2\ln 2 - 1 = 0.386294\ldots$ bits (donc pas tout à fait 0.5, comme on aurait pu le croire). Pour $b = 3$ et $b = 10$, nous trouvons des gains espérés de $0.323959\ldots$ et $0.173158\ldots$ chiffres, respectivement.

Nous observons donc que cette limite est *décroissante* en b, et que le b qui nous donne la meilleure espérance de gain est $b = 2$!

Codage et décodage

Sachant

- $n = b^k + r$ avec $0 \leqslant r < b^k(b-1)$,
- n est t. q. $b-1 \mid n-1$,
- le seuil $s = b^k - \frac{r}{b-1}$.

Pour encoder une valeur v,

- Si $v < s$, c'est un code court et nous émettons v sur k chiffres ;
- Si $v \geqslant s$, c'est code long et nous émettrons un code sur $k+1$ chiffres.

Les codes pour les valeurs de 0 à $s-1$ étant courts, le premier code long est la valeur s décalée d'un chiffre, c'est-à-dire sb. Comme les valeurs $v \geqslant s$ sont codées relativement au premier code long, nous émettons $sb + (v-s) = v + s(b-1)$ sur $k+1$ chiffres.

Codage

fonction *Encoder*(v,n,b)
  $k = \lfloor \log_b n \rfloor$
  $r = n - b^k$
  $s = b^k - \frac{r}{b-1}$
  si $v < s$ **alors**
    émettre v sur k chiffres
  sinon
    émettre $v + s(b-1)$ sur $k+1$ chiffres

décodage

fonction *Décoder*(v,n,b)
  $k = \lfloor \log_b n \rfloor$
  $r = n - b^k$
  $s = b^k - \frac{r}{b-1}$
  $v \leftarrow$ lire k chiffres
  si $v < s$ **alors**
    retourner v
  sinon
    $v \leftarrow vb+$(lire 1 chiffre)
    retourner $v - s(b-1)$

Ce qu'il faut retenir

- Les codes économiques existent depuis longtemps, mais n'ont jamais été très bien étudiés ;
- Nous avons généralisé les codes économiques à une base $b \geqslant 2$ quelconque ;
- Nous avons généralisé des résultats précédemment limités au cas particulier $b = 2$;
- Nous avons présenté de nouveaux résultats, comme le gain espéré.

Références

[1] Willard L. Eastman — *US Patent 5,087,913 : Short-Record Data Compression and Decompression System* — US Patent Office (février 1992).

[2] Willard L. Eastman, Abraham Lempel, Jacob Ziv, Martin Cohn — *US Patent 4,464,650 : Apparatus and Method for Compressing Data Signals and Restoring the Compressed Data Signals* — US Patent Office (août 1984).

[3] David A. Huffman — *A Method for the Construction of Minimum-Redundancy Codes* — Procs of the IRE, vol. 49, n° 9 (septembre 1952), p. 1098–1101.

[4] Steven Pigeon — *Contributions à la compression de données* — Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, (décembre 2001).

[5] Hidetoshi Yokoo — *An Improved Ziv-Lempel Coding Scheme for Universal Source Coding* — Electronics and Communications in Japan, Part I, vol. 69, n° 12 (1986), p. 12–20.


Ô NOBLE VIRGILE, QUI SONT CES GENS?
CE SONT CEUX QUI LAISSAIENT LA PREUVE AUX LECTEURS
AH OUAIS? BIEN FAIT.