Codes économiques réexaminés : généralisations, bornes et algorithmes

Steven Pigeon

Département de mathématiques, informatique et génie
Université du Québec à Rimouski
300 Allée des Ursulines, Rimouski, Québec
steven_pigeon@uqar.ca

Résumé

Présentés par Eastman et al., les codes économiques permettent d'encoder des valeurs $0 \leqslant v < n$, indépendamment identiquement uniformément distribuées lorsque n n'est pas une puissance de deux. Bien que d'apparence simple, ce cas particulier des codes de Huffman n'a jamais été complètement caractérisé. Dans cette communication, nous généralisons les codes économiques aux bases $b \geqslant 2$, donnons les longueurs espérées des codes, leur inefficacité maximale ainsi que le nombre espéré de chiffres sauvés. Finalement, nous présenterons les procédures d'encodage et de décodage.

Mots clefs

Codes de Huffman, codes économiques, compression de données, distribution uniforme.

1 Introduction

Il n'est pas rare que nous devions encoder des entiers $0 \leqslant v < n$, où n n'est pas une puissance de deux et pour lesquels nous n'avons pas de bonnes raisons de croire qu'ils soient tirés d'une distribution nettement différente de la distribution uniforme. La méthode naïve dans ce cas est d'utiliser un nombre fixe de bits, $\lceil \log_2 n \rceil$, pour encoder les valeurs. Cette approche est rapide, mais peu efficace, car lorsque $2^k < n \ll 2^{k+1}$, nous gaspillons presqu'un bit. Une meilleure solution serait d'utiliser un code de longueur variable, mais aussi simple que possible. Les codes économiques, présentés par Eastman *et al.*, exploitent un cas particulier des codes de Huffman pour résoudre ce problème [1]. Cependant, bien que mentionnés à plusieurs reprises dans la littérature (par ex. [2, 3, 4, 5]), ces codes n'ont jamais été décrits avec précision.

Dans cet article, nous décrivons la généralisation de ces codes pour une base $b \geqslant 2$. Certaines expressions, déjà connues pour le cas particulier $b = 2$, comme la longueur espérée des codes [3] et l'inefficacité maximale [5] seront étendues à toutes les bases. Nous abordons aussi de nouvelles considérations comme le gain spécifique et le gain espéré, qui n'ont jamais été caractérisés. Finalement, nous discuterons des procédures d'encodage et de décodage.

2 Codes économiques généralisés

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

2.1 Structure des codes

Pour coder optimalement $b^k \leqslant n < b^{k+1}$ valeurs distinctes, il nous faut construire un arbre de codes complet où les feuilles se trouvent, au plus, sur deux profondeurs adjacentes (voir fig. 1). Pour ce faire, nous réutiliserons la procédure de Huffman [6]. Pour la construction l'arbre de codes b-aire complet pour n valeurs, nous utiliserons la décomposition

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

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

Nous savons que si $r \neq 0$, l'arbre de code ne pourra pas être parfait, car un arbre parfait aurait exactement b^k feuilles ; or, il nous en faut $b^k + r$. Donc certaines feuilles seront à la profondeur k, recevant des codes courts, tandis que les autres seront à la profondeur $k+1$, recevant des codes longs.

Pour utiliser la procédure de Huffman, c'est-à-dire itérativement fusionner les b feuilles de poids plus faibles pour finir par n'obtenir qu'un seul nœud actif, la racine, nous devons avoir n tel que $b-1 \,|\, n-1$. Il se peut donc que nous ayons à ajuster n pour accommoder cette condition, en posant $n' = n + (n-1) - (n \mod b-1)$.

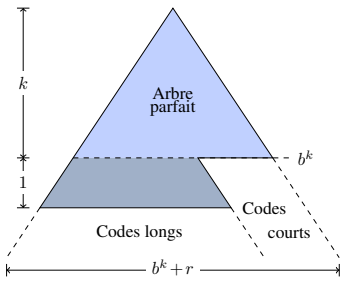
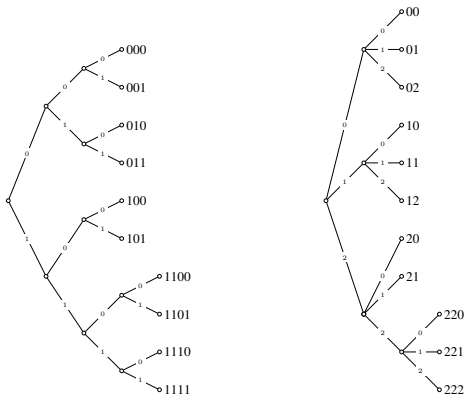


FIGURE 1 – *La structure du code.*

Pour créer l'arbre de codes, nous commençons avec n



(a) *Code économique binaire pour* $n = 10$.

(b) *Code économique ternaire pour* $n = 10$ *(promu à* 11*).*

FIGURE 2 – *Exemples de codes économiques.*

nœuds, tous de poids égal. En utilisant la procédure de Huffman, 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. Le nombre m de fusions nécessaires pour atteindre un arbre parfait est tel que $r = m(b-1)$, soit encore $m = \frac{r}{b-1}$. Puisque ces m fusions représentent chacune b feuilles, nous aurons $l = bm = \frac{br}{b-1}$ codes longs et $s = n - bm = b^k - \frac{r}{b-1}$ codes courts. Cette valeur s nous servira de seuil pour les procédures rapides d'encodage et de décodage.
La fig. 2 donne deux exemples de codes économiques pour $n = 10$. À la fig. 2a, nous trouvons un code binaire pour $n = 2^3 + 2$, avec un seuil $s = 2^3 - 2 = 6$, tandis qu'à la fig. 2b, nous devons promouvoir n à 11 pour accommoder le code ternaire, avec un seuil de $s = 3^2 - 1 = 8$.

2.2 Longueur espérée

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

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

Comme $0 \leqslant r < b^k(b-1)$, nous pouvons vérifier que la longueur moyenne est $k \leqslant \bar{L}_B(n) < k+1$.

2.3 Inefficacité maximale

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, toujours en supposant les valeurs uniformément iid. Avec un code économique, nous avons

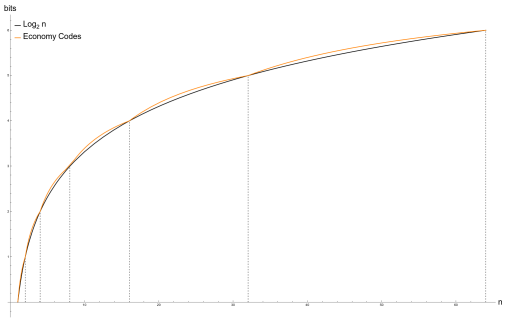


FIGURE 3 – *Longueur des codes économiques pour le cas* $b = 2$ *vs* $\log_2 n$.

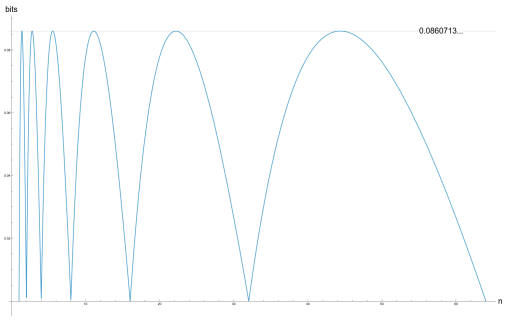


FIGURE 4 – *L'inefficacité* $\bar{L}_b(n) - \log_b n$ *des codes économiques pour le cas* $b = 2$.

$\bar{L}_b(n) \geqslant \log_b n$ avec l'égalité seulement lorsque $r = 0$, et une différence $0 < \bar{L}_b(n) - \log_b n < 1$ autrement. La fig. 3 montre la longueur moyenne d'un code économique pour le cas particulier $b = 2$ *vs* un code idéal (et hypothétique) utilisant exactement $\log_2 n$ bits, tandis que la fig. 4 montre l'inefficacité $\bar{L}_b(n) - \log_b n$ des codes économiques.
Nous remarquons que $\bar{L}_b(n) - \log_b n$ est convexe en $0 \leqslant r < b^k(b-1)$, nous pouvons donc trouver le maximum de l'inefficacité en résolvant

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(\bar{L}_b(n) - \log_b(n)\right) = 0 \tag{2}$$

pour r. Nous trouvons que

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(\bar{L}_b(n) - \log_b(n)\right) = \frac{b^{k+1}\ln b - (b-1)(b^k + r)}{(b-1)(b^k + r)^2 \ln b},$$

puis nous résolvons

$$\frac{b^{k+1}\ln b - (b-1)(b^k + r)}{(b-1)(b^k + r)^2 \ln b} = 0 \tag{3}$$

pour r. L'éq. (3) est satisfaite lorsque son numérateur est nul, c'est-à-dire lorsque

$$b^{k+1}\ln b = (b-1)(b^k + r), \tag{4}$$

et nous trouvons que le r qui maximise l'inefficacité est

$$r^* = \frac{b^k(b(\ln b - 1) + 1)}{b-1}. \tag{5}$$

En substituant r^* dans $n = b^k + r$, nous trouvons que le

« pire » $b^k < n < b^{k+1}$ est donné (approximativement) par

$$n^* = b^k + r^* = \frac{b^{k+1}\ln b}{b-1}. \tag{6}$$

Maintenant, en utilisant les éqs. (5) et (6), nous trouvons

$$\begin{aligned}\bar{L}_b(n^*) - \log_b n^* &= \frac{br^*}{(b-1)(b^k+r^*)} - \log_b\left(1+\frac{r^*}{b^k}\right)\\ &= \frac{1}{b-1} + \frac{\ln(b-1)-\ln\ln b-1}{\ln b}.\end{aligned} \tag{7}$$

Pour le cas particulier $b=2$, l'éq. (7) nous donne

$$\begin{aligned}&\frac{1}{b-1} + \frac{\ln(b-1)-\ln\ln b-1}{\ln b}\\ &\quad= \frac{1}{2-1} + \frac{\ln(2-1)-\ln\ln 2-1}{\ln 2}\\ &\quad= 0.086071332\ldots,\end{aligned}$$

un résultat précédemment obtenu [5, p. 102–104], et qui peut être vu comme un cas particulier du résultat de Gallager pour une distribution quelconque [7]. Pour d'autres bases, comme par exemple, $b=3$ et $b=10$, nous trouvons que l'inefficacité maximale est de $0.135084\ldots$ et $0.268843\ldots$ chiffres, respectivement. Un examen plus attentif de l'éq. (7) nous révèle que l'inefficacité est strictement croissante en b et bornée supérieurement par 1 lorsque b tends vers l'infini. Cela nous permet de conclure que la base avec la plus petite inefficacité maximale est la base $b=2$.

2.4 Gain spécifique et gain espéré

Pour un code économique en base b pour $n = b^k + r$, avec $0 < r < b^k(b-1)$, nous sauverons, en moyenne

$$\begin{aligned}\lceil\log_b n\rceil - \bar{L}_b(n) &= (k+1) - \left(k + \frac{br}{(b-1)(b^k+r)}\right)\\ &= 1 - \frac{br}{(b-1)(b^k+r)}\end{aligned} \tag{8}$$

chiffres par rapport à un code fixe qui utilise toujours $\lceil\log_b n\rceil = k+1$ chiffres.
Le gain spécifique donné par l'éq. (8) est maximal lorsque $r>0$ est petit et presque nul lorsque $r=(b^k-1)(b-1)$, soit $n \approx b^{k+1}$. Le cas $r=0$ correspond à $n=b^k$, auquel cas le code de taille fixe est optimal. Toutefois, puisque nous devons avoir $b-1 \mid n-1$, le plus petit $r \neq 0$ n'est pas forcément 1 : le plus petit $r \neq 0$ est $r = b-1$. En conséquence, le gain maximal est de

$$\begin{aligned}1 - \frac{br}{(b-1)(b^k+r)} &= 1 - \frac{b(b-1)}{(b-1)(b^k+(b-1))}\\ &= 1 - \frac{b}{b^k+b-1}\end{aligned}$$

chiffres. Les codes économiques sont les plus intéressants lorsque $b^k < n \ll b^{k+1}$, car nous sauverons presqu'un chiffre à chaque code. Par contre, les codes économiques le sont beaucoup moins lorsque n est presque b^{k+1}. Si nous nous questionnons sur l'utilité d'un code économique, nous pouvons établir un seuil d'utilité $0 < \alpha < 1$ tel que si

$$1 - \frac{br}{(b-1)(b^k+r)} \geqslant \alpha,$$

ou, de façon équivalente, si

$$r \leqslant b^k \frac{(1-\alpha)(b-1)}{1+\alpha(b-1)},$$

nous jugerons utile l'emploi de codes économiques. Sinon, nous utiliserons simplement un code fixe sur $k+1$ chiffres. Quel est le gain espéré étant donné un $b^k < n < b^{k+1}$ uniformément distribué sur cet intervalle ? Bien qu'Eastman *et al.* nous assurent que nous sauvons en moyenne un demi bit [1, p. 18], voyons ce qu'il en est vraiment. L'espérance de l'éq. (8), en ne considérant que les $b^k < n < b^{k+1}$ valides, c'est-à-dire tels que $b-1 \mid n-1$, est

$$\begin{aligned}&\mathbb{E}\left[\lceil\log_b n\rceil - \bar{L}_b(n)\right]\\ &\quad= \frac{1}{b^k-1} \sum_{\substack{r=b-1,2(b-1),\\ \ldots,(b^k-1)(b-1)}} 1 - \frac{br}{(b-1)(b^k+r)}\\ &\quad= 1 - \frac{b}{(b^k-1)(b-1)} \sum_{j=1}^{b^k-1} \frac{j(b-1)}{b^k+j(b-1)}.\end{aligned} \tag{9}$$

La forme $\sum_{r=1}^{u} \frac{r}{v+r}$ doit être transformée pour être exploitable. 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. Poursuivons l'éq. (9) :

$$\begin{aligned}&\mathbb{E}\left[\lceil\log_b n\rceil - \bar{L}_b(n)\right]\\ &\quad= 1 - \frac{b}{(b^k-1)(b-1)} \sum_{j=1}^{b^k-1} \frac{j(b-1)}{b^k+j(b-1)}\\ &\quad= 1 - \frac{b}{(b^k-1)(b-1)} \times\\ &\qquad\left((b^k-1) - \frac{b^k}{b-1}\left(H\left(\frac{b^{k+1}}{b-1}-1\right) - H\left(\frac{b^k}{b-1}\right)\right)\right)\\ &\quad= \frac{b^{k+1}}{(b^k-1)(b-1)^2}\left(H\left(\frac{b^{k+1}}{b-1}-1\right) - H\left(\frac{b^k}{b-1}\right)\right) - \frac{1}{b-1}.\end{aligned} \tag{10}$$

Si l'éq. (8) nous donne le nombre de chiffres sauvés en

moyenne pour un n donné, l'éq. (10) nous donne le nombre de chiffres que nous pouvons espérer d'un n valide entre deux puissances de b. Sachant

$$\lim_{k\to\infty} \mathbb{E}\left[\lceil \log_b n \rceil - \bar{L}_b(n)\right] = \frac{b\ln b-(b-1)}{(b-1)^2}, \qquad (11)$$

nous pouvons trouver le gain espéré en fonction de b. Pour $b=2$, l'éq. (11) nous donne un gain espéré de $2\ln 2-1=0.386294\ldots$ bits, c'est-à-dire que pour un n quelconque, nous pouvons nous attendre à sauver environ 0.39 bits — pas un demi bit ! Pour $b=3$ et $b=10$, nous trouvons des gains espérés de $0.323959\ldots$ et $0.173158\ldots$ chiffres, respectivement. De plus, nous voyons que l'éq. (11) tend vers zéro alors que b croît, donc que le gain espéré est le plus grand lorsque b est le plus petit, c'est-à-dire lorsque $b=2$.

2.5 Codage et décodage

Les procédures d'encodage et de décodage sont simples et rapides, exploitant la structure des codes pour nous dispenser de la construction explicite de l'arbre de codes.
Revenons à la fig. 1. Encore une fois, utilisons la décomposition $n=b^k+r$ avec $0\leqslant r<b^k(b-1)$ et n tel que $b-1\,|\,n-1$. Les codes courts, au nombre de $s=b^k-\frac{r}{b-1}$, utiliseront k chiffres, tandis que les codes longs en utiliseront $k+1$.
Pour encoder une valeur v, si $v<s$, la valeur correspond à un code court et nous émettons le code « naturel » pour v sur k chiffres. Si $v\geqslant s$, la valeur correspond à un 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.
Pour décoder, nous lisons d'abord k chiffres et vérifions si cette valeur est inférieure à s. Si c'est le cas, nous retournons la valeur lue telle quelle. Sinon, il s'agit des k premiers chiffres d'un code long ; il nous faut lire un chiffre supplémentaire qu'on ajoute aux chiffres déjà lus, formant la valeur v. Pour inverser l'encodage, nous retournons $v-s(b-1)$.
Les procédures sont montrées aux algorithmes 2.1 et 2.2. Notons qu'une implémentation optimisée s'abstiendrait de recalculer la décomposition. De plus, pour le cas spécial $b=2$, la méthode se simplifie grandement puisque $b-1$ devient 1.

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

Algorithme 2.1 – *Procédure d'encodage.*

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 $v-s(b-1)$

Algorithme 2.2 – *Procédure de décodage.*

3 Conclusion

Nous avons présenté une généralisation des codes économiques d'Eastman *et al.* aux bases $b\geqslant 2$. Nous avons donné les expressions pour la longueur espérée et pour l'inefficacité maximale. Nous avons montré comment généraliser le gain spécifique d'un code économique pour un n particulier au gain que l'on peut espérer d'un $b^k<n<b^{k+1}$ quelconque, mais toujours tel que $b-1\,|\,n-1$. Nous avons montré que l'inefficacité maximale pour le cas particulier $b=2$ est d'au plus $0.086071\ldots$ bits, alors que le gain espéré est de $0.396294\ldots$ bits, et que le gain espéré est le plus grand lors que $b=2$. Finalement, nous avons montré que les codes économiques sont à la fois très simples et très efficaces, du moins si nous supposons que les valeurs à coder sont tirées d'une distribution uniforme.

Références

[1] Willard L. Eastman, Abraham Lempel, Jacob Ziv, et Martin Cohn. US Patent 4,464,650 : Apparatus and Method for Compressing Data Signals and Restoring the Compressed Data Signals, Août 1984.

[2] Willard L. Eastman. US Patent 5,087,913 : Short-Record Data Compression and Decompression System, Février 1992.

[3] Hidetoshi Yokoo. An Improved Ziv-Lempel Coding Scheme for Universal Source Coding. *Electronics and Communications in Japan, Part I*, 69(12) :12–20, 1986.

[4] Peter Tischer. A Modified Lempel-Ziv-Welch Data Compression Scheme. *Australian Computer Science Communications*, 9(1) :262–272, 1987.

[5] Steven Pigeon. *Contributions à la compression de données*. Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, Décembre 2001.

[6] David A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Procs of the IRE*, 49(9) :1098–1101, Septembre 1952.

[7] Robert G. Gallager. Variations on a Theme by Huffman. *IEEE Trans on Information Theory*, 24(6) :668–674, Novembre 1978.